IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF WISCONSIN

	)	
WISCONSIN ALUMNI RESEARCH FOUNDATION	)	
	)	
Plaintiff,	)	
	)	
v.	)	**Case No. 08-C-78-C**
	)	
INTEL CORPORATION	)	
	)	
Defendant.	)	
	)	

PLAINTIFF'S RESPONSE TO DEFENDANT INTEL CORPORATION'S OPENING MARKMAN BRIEF

TABLE OF CONTENTS

TABLE OF AUTHORITIES

INTRODUCTION

Intel attempts to limit all of the claims of the ’752 patent[1] to one of the central observations that led to the discovery of ’752 invention – namely the observation that it was generally only a few LOAD/STORE instruction pairs that could not be executed out of order, (i.e., where the LOAD is executed before the STORE even though the LOAD occurs after the STORE in the computer program). Umberger Decl. Ex. 1, Col. 3, ll. 51-57. However, as is frequently the case, the claims of the ’752 patent are not all directed to that aspect of the invention. Subsumed within the inventors’ work, for example, was also the recognition that LOAD instructions that had been successfully executed out of order in the past (i.e., LOAD instructions that were executed ahead of the STORE instructions even though the LOAD instructions occur after the STORE instructions) generally would do so in the future. The utility of this observation could be obtained simply by predicting for a given LOAD instruction, based on its past behavior, whether one should speculatively execute that LOAD instruction out of order in this particular instance. One could speculate if the likelihood of mis-speculation was low; otherwise not. This is the subject matter to which Claim 1 of the ’752 patent is directed. Moreover, as the specification makes clear, this feature of the invention was not limited to specific LOAD/STORE instruction pairs, as Intel argues.

Because Intel places so much emphasis on the LOAD/STORE pair aspect of the invention, it is worth preliminarily looking at the structure of the claims of the ’752 patent. That review will demonstrate that the added utility that comes from actually keeping track of

[1] A copy of the patent in suit, U.S. Patent No. 5,781,752 (“the ‘752 patent”), is attached as Exhibit 1 to the Declaration of Michelle M. Umberger in Support of Plaintiff’s Motion for Construction of Claims and Opening Brief in Support of Motion (hereinafter “Umberger Decl.”) (Dkt. No. 30).

LOAD/STORE pairs is contained in the later claims that are dependent on Claim 1. In addition, that review of the claims will also demonstrate that the doctrine of claim differentiation undercuts Intel's attempt to limit Claim 1 to features added in the dependent claims.

The claims are written in multiple tiers that are collectively geared towards delaying a LOAD instruction when necessary and towards releasing the delayed LOAD instruction for processing as soon as possible after the instruction has been delayed. Specifically, in the ***first tier*** of the '752 invention, if a LOAD instruction does not have a history of mis-speculation (i.e., a history of conflicts with STORE instructions), it is executed with the expectation that a lack of conflicts in the past indicates a lack of conflict in the future. In the ***second tier*** of the invention (which is captured in Claim 1), a LOAD instruction is delayed (i.e., not executed in a data speculative manner) when a prediction associated with the LOAD indicates a high likelihood of a conflict with a STORE instruction. The delayed LOAD instruction is executed when it can be confirmed that the LOAD does not conflict with any of the pending STORE instructions. Thus, the *second tier* offers a solution that prevents conflicts (or data mis-speculations) between a LOAD instruction and **all** pending STORE instructions.

An extension to this scheme (which is captured in dependent Claim 5 and forms the ***third tier*** of the invention) is triggered when a STORE that is "highly likely" to conflict with the delayed LOAD instruction executes. In that event, the delayed LOAD is allowed to proceed with the expectation that it is "highly likely" that the data dependence has been resolved due to the execution of that STORE. To facilitate this tier, the '752 patent teaches a prediction table that maintains pairs of LOAD and STORE instructions that are "highly likely" to conflict. Thus, a delayed LOAD instruction is speculatively released when the STORE instruction most likely to be the cause of problems has been executed, but before it can be ascertained that there are no conflicts with any other additional pending STORE instructions.

Intel seeks to limit the entire ’752 invention to that extension (i.e, the *third tier*). According to Intel, the ’752 invention is limited to preventing mis-speculation only between a LOAD and a STORE of the identified LOAD/STORE pairs, i.e., a LOAD instruction is delayed only when it appears with its paired STORE instruction.[2]

In summary, the crux of the dispute is whether the broad language of claim 1 of the ’752 patent requires that a LOAD instruction be delayed only in regard to the paired STORE instruction or whether a LOAD instruction is delayed without regard to the occurrence of the paired STORE instruction. The specification of the ’752 patent indisputably discloses that the decision to delay a LOAD instruction is **not** based on whether the paired STORE instruction appears. The decision is based solely on the prediction associated with the LOAD instruction. *See* Sections I.A.-B., *infra*. In fact, the specification expressly contemplates circumstances when a paired STORE instruction is not even present, yet the LOAD is nonetheless speculatively executed. The ’752 patent contemplates this scenario only because the decision to delay a LOAD instruction is independent of whether the paired STORE appears or not. *See* Section I.C., *infra*.

Based on an inaccurate description of the ’752 invention as operating solely on a LOAD/STORE pair basis, Intel asks the Court to import the “load/store pair” limitation into claims that are devoid of any such requirement. *See* Sections I.D.-G., *infra*. Because Intel’s

[2] Intel’s interpretation of how the *third tier* operates is partially incorrect. Intel believes that the delayed LOAD is executed in a non-speculative manner after the paired STORE appears. However, as discussed in Section I.C. below, in the *third tier*, after the paired STORE executes, the delayed LOAD is executed in a data speculative manner because the LOAD may still conflict with any of the other pending STORE instructions.

understanding of the ’752 invention is flawed, a cascade of errors ripples through its proposed constructions:[3]

Data speculation circuit: Intel incorrectly limits the operation of this circuit to detecting and preventing mis-speculation between “load/store pairs.” Instead, the circuit detects mis-speculation generally between a LOAD and all prior STORE instructions. *See* Section I, *infra.*

Limitation (b) of Claim 1: Intel argues that the “instructions” referred to in Claim 1 as being delayed by the “prediction threshold detector” are a “load/store pair.” The correct interpretation is, as the plain language indicates, that the delayed instructions are the LOAD instructions. *See* Section II, *infra*.

Prediction: Similarly, Intel argues that “prediction” is the likelihood that speculative execution of a “load/store pair” will result in a conflict because the ’752 invention is limited to preventing mis-speculation between LOAD/STORE pairs. Again, Intel’s premise is flawed, and further its proposed construction of “prediction” ignores the plain language of the ’752 patent. As used in Claim 1, the term “prediction” forecasts the likelihood of a data mis-speculation by a LOAD instruction (i.e., Claim 1 recites that the “predictor” “produce[s] a prediction associated with the particular data consuming instruction [LOAD],” with no reference to the STORE instruction). *See* Sections III-V, *infra*.

Intel’s proposed constructions for other terms are equally divorced from the specification of the ’752 patent. For example, even though the preamble of Claim 1 expressly limits mis-speculation to a ***data*** mis-speculation, Intel’s definition captures both *control* and *data* mis-

[3] Intel has moved for the construction of several claim terms that were not included in WARF’s motion and opening brief. While WARF does not oppose Intel’s motion for construction of these terms (or Intel’s request for a hearing), WARF disagrees with the substance of Intel’s constructions, and herein sets forth its own proposed constructions for the terms raised by Intel.

speculation in its ambit. *See* Sections VI-VII, *infra*. Similarly, by ignoring the context in which the term <u>in fact executed</u> appears in the preamble of Claim 1, Intel proposes an incomplete construction of that term. *See* Section VIII, *infra*.

CONSTRUCTION OF DISPUTED TERMS

I. "DATA SPECULATION CIRCUIT"

The disputed phrase "data speculation circuit" is a good vehicle to examine the role that LOAD/STORE pairs play in the '752 patent. Apparently Intel agrees, having devoted a substantial portion of its opening brief to this phrase. The phrase appears in Claim 1 as follows:

> 1. In a processor capable of executing program instructions in an execution order differing from their program order, the processor further having a <u>**data speculation circuit**</u> for detecting data dependence between instructions and detecting a mis-speculation where a [LOAD instruction] dependent for its data on a [STORE instruction] of earlier program order, is in fact executed before the [STORE instruction], a data speculation decision circuit comprising:
>
> a) a predictor receiving a mis-speculation indication from the data speculation circuit to produce a prediction associated with the [LOAD instruction] and based on the mis-speculation indication; and
>
> b) a prediction threshold detector preventing data speculation for instructions having a prediction within a predetermined range.

Umberger Decl. Ex. 1, Claim 1. The parties have proposed the following constructions:

Claim Term	**WARF's Proposal**	**Intel's Proposal**
data speculation circuit	a circuit that detects data dependence between LOAD and STORE instructions and tracks execution of such instructions in order to detect any mis-speculations arising from the data speculative execution of LOAD instructions	a circuit that detects data dependence between load/store pairs and detects a mis-speculation by a load instruction that is in fact executed

Superficially, the two proposed constructions may not appear to be that far apart. The parties agree that the invention disclosed in the '752 patent identifies a conflict between a LOAD instruction and a STORE instruction (i.e., it takes two to mis-speculate, a STORE and a LOAD).

However, the parties disagree as to how the invention recited in Claim 1 of the ’752 patent prevents mis-speculation once such a conflict is identified. In light of the specification, Claim 1 is clear: once a mis-speculation behavior of a LOAD instruction is identified, data speculation by that LOAD is prevented **as to <u>all</u> prior STORE instructions** (i.e., the LOAD instruction will not execute until it is confirmed that the LOAD does not conflict with any prior STORE).[4] Intel’s justification for importing the “load/store pair” limitation from the specification into Claim 1 is based on its mis-characterization of the ’752 invention as operating on a “load/store” pair basis. It attempts to limit the invention (and Claim 1 as consequence) to require that once a conflicting LOAD/STORE pair is identified, speculation of the <u>particular LOAD</u> instruction is prevented only as to that <u>particular STORE</u> instruction. In its opening brief, Intel repeatedly incants this formulation of the ’752 invention:

- “The purpose of the ’752 patent is to detect data dependence between a **<u>particular</u>** load and a particular store (a “pair”), so that data mis-speculation by that particular pair can be prevented. Defendant Intel Corporation’s Opening Markman Brief (“Intel Markman Br.”) at 32 (emphases in original).

- “the purpose of the invention is . . . to detect—and thereby ultimately prevent—mis-speculation by a load instruction <u>when it appears with a particular store instruction</u>.” Intel Markman Br. at 32 (emphasis in original).

- “the specification states that the animating idea of the (alleged) invention is to identify the load/store pairs that mis-speculate, and then prevent **<u>them</u>** from speculating” Intel Markman Br. at 29 (emphasis added).

- “Every figure and every description is directed towards identifying and disabling a mis-speculating load instruction **<u>that is paired with a particular store instruction</u>**.” Intel Markman Br. at 31 (emphasis in original).

[4] The *third tier* (which is reflected in dependent claims) builds upon where the *second tier* leaves. The *second tier* decides whether to delay a LOAD and the *third tier* decides when to release that delayed LOAD. If the paired STORE does not occur, the delayed LOAD is released when all pending STORE instructions confirm that they do not conflict with that LOAD. Umberger Decl. Ex. 1, Col. 10, ll. 26-31.

Contrary to Intel's assertions (and as will be shown below), the specification does not support, much less compel, this narrow definition of the '752 invention and of the otherwise broad phrase "data speculation circuit." The plain language of Claim 1 nowhere uses the term "pair" (despite the use of that term in the specification) nor otherwise limits the circuit to one that prevents speculation by a "load/store pair." Intel's construction is thus contrary to the Federal Circuit's oft-repeated admonition that "it is important not to import into a claim limitations that are not a part of the claim." *Superguide Corp. v. DirecTV Enters.*, 358 F.3d 870, 875 (Fed. Cir. 2004). "Where a specification does not *require* a[n extraneous] limitation, that limitation should not be read from the specification into the claims." *Anheuser-Busch Cos. v. Crown Cork & Seal Techs. Corp.*, 121 Fed. Appx. 388, 393 (Fed. Cir. 2004) (quoting *E.I. Du Pont de Nemours & Co. v. Phillips Petroleum Co.*, 849 F.2d 1430, 1433 (Fed. Cir. 1988)).

A. The '752 Patent Provides A Generic Solution For LOAD Instructions Based On The LOAD/STORE Locality Phenomenon

The Summary of the '752 patent begins by pointing out that the inventors realized that (1) only a few LOAD/STORE pairs were causing a majority of the data mis-speculations (Umberger Decl. Ex. 1, Col. 3, ll. 51-53); and (2) if a LOAD instruction conflicted with a STORE, then that LOAD was "highly likely" to conflict with that STORE again (Umberger Decl. Ex. 1, Col. 3, ll. 54-57). These two observations led the inventors to conceive a broad invention that generally seeks to delay the LOAD instructions that are causing the majority of data mis-speculations and then seeks to release the delayed LOAD instructions as soon as possible. Thus, it is not surprising that the '752 specification makes frequent references to the LOAD/STORE instruction pairs. Umberger Decl. Ex. 1, Col. 11, ll. 29-33; Col. 14, ll. 1-3. However, as shown below, though the embodiments in the '752 patent rely on LOAD/STORE pairs and certain dependent claims (e.g., Claim 5) specifically claim such embodiments, Claim 1 (and the data speculation circuit) is not limited to detecting and preventing mis-speculation between those LOAD/STORE pairs.

To keep track of the conflicting LOAD/STORE instruction pairs, the example disclosed in the '752 patent maintains a prediction table. Umberger Decl. Ex. 1, Col. 3, ll. 57-62. The prediction table (see Fig. 5 below) contains a LOAD instruction, a STORE instruction and a prediction (109) that indicates the likelihood of a conflict between the STORE and the LOAD instruction. Umberger Decl. Ex. 1, Col. 11, ll. 8-14 and 29-33. In the disclosed embodiment, the prediction is used to decide whether to delay a LOAD instruction. *Id.* at Col. 3, l. 1 - Col. 4, l. 5; Col. 4, ll. 41-47; Col. 11, ll. 29-41.

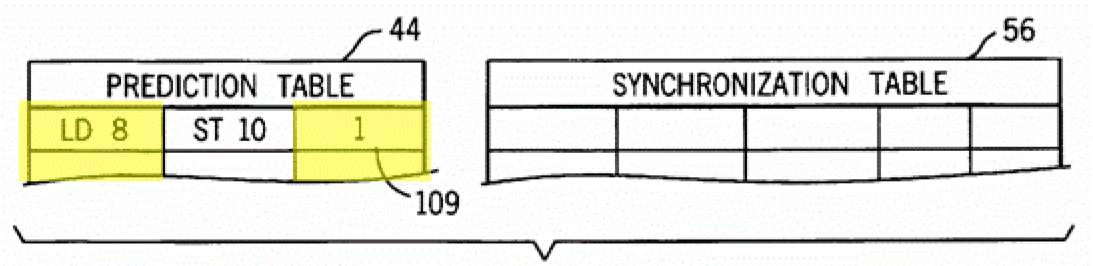


FIG. 5

PREDICTION TABLE

The '752 patent also discloses a synchronization table, which is employed after a LOAD instruction has been delayed. Umberger Decl. Ex. 1, Col. 11, ll. 37-47. Delaying a LOAD instruction implies that the delayed LOAD (a data consuming instruction) is "highly likely" to use the data from a STORE (a data producing instruction), i.e., the delayed LOAD may benefit from synchronizing with the STORE. *Id.* at Col. 11, ll. 26-33. The synchronization table indicates whether there is in fact a pending LOAD instruction awaiting its dependent STORE instruction. *Id.* at Col. 11, ll. 42-47.

The inventors specifically contemplated that the ’752 invention could be selectively implemented in a three-tiered manner where each tier would utilize different portions of the prediction and the synchronization table. Umberger Decl. Ex. 1, Col. 3, l. 64 - Col. 4, l. 8. The claims of the ’752 patent mirror these tiers. Each of these tiers is driven by a decision associated with the execution of LOAD instructions:

<u>First tier</u>: The processor checks whether the LOAD instruction has mis-speculated in the past. If the LOAD instruction does not have a history of mis-speculation, it is executed speculatively. Umberger Decl. Ex. 1, Col. 3, ll. 63-67; Col. 4, ll. 48-53; Col. 11, ll. 19-24.

<u>Second tier</u>: If the LOAD instruction has a history of mis-speculation, then the second tier is implicated. The processor checks the prediction associated with the LOAD instruction. And, the processor delays the LOAD instruction if the prediction indicates so. Umberger Decl. Ex. 1, Col. 3, l. 67 – Col. 4. l. 5; Col. 4, ll. 41-47. In the disclosed embodiment, only the first column (which holds the LOAD) and the third column (which holds the associated prediction) (highlighted in yellow in the figure above) are utilized when deciding whether to delay or execute the LOAD. *Id.* at Col. 11, ll. 26-40. Independent Claim 1 captures this tier of the invention.

<u>Third tier</u>: The *third tier* picks up from where the *second tier* leaves. The *second tier* identifies which LOAD instructions should be delayed and the *third tier* subsequently decides *when* to release certain of those delayed LOAD instructions. Umberger Decl. Ex. 1, Col. 4, ll. 5-7. The STORE paired with the LOAD is the STORE that is “highly likely” to conflict with the LOAD. *Id.* at Col. 11, ll. 29-33. If and when that STORE executes, the delayed LOAD can be allowed to proceed in a data speculative manner with the expectation that the dependency between the two has been resolved. *Id.* at Col. 12, ll. 50-58. However, it is only “highly likely”

that the dependency has been resolved and the released LOAD can still conflict with other pending STORE instructions. *Id.* at Col. 10, ll. 25-31, 27-45; Col. 12, ll. 56-58.

To enable this third tier, after the processor decides to delay a LOAD, it places the LOAD and the paired STORE in the synchronization table to indicate that a pending LOAD instruction is awaiting its dependent STORE instruction. Umberger Decl. Ex. 1, Col. 11, ll. 48-62. When the dependent STORE executes at a later instant of time, it finds the pending LOAD in the synchronization table and releases it. *Id.* at Col. 12, ll. 50-58. Claim 1 does not recite this aspect of the invention. The dependent claims, e.g., Claim 5, wherein the "synchronization table" is first recited, capture this aspect of the invention.[5]

Intel seeks to limit the '752 invention to an aspect of the invention (i.e., the *third tier*) that releases the delayed LOAD instruction when the paired STORE instruction executes. Intel's efforts are to no avail because a patentee is not even limited to claiming only the preferred embodiment let alone an aspect of an embodiment, as "the fact that the inventor may have anticipated that the invention would be used in a particular way does not mean that the scope of the invention is limited to that context." *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 909 (Fed. Cir. 2004); *see also* Umberger Decl. Ex. 1, Col. 5, ll. 30-33 ("Such embodiment does not necessarily represent the full scope of the invention, however, and reference must be made therefore to the claims for interpreting the scope of the invention."); Umberger Decl. Ex. 1, Col. 14, ll. 33-35 ("In order to apprise the public of the various embodiments that may fall within the scope of the invention, the following claims are made . . . ."). Accordingly, the "court will not import limitations into claims from examples or embodiments appearing only in a patent's

---

[5] In the disclosed embodiment, besides the synchronization table, the *third tier* utilizes the first and second columns of the prediction table. Umberger Decl. Ex. 1, Col. 11, ll. 37-41.

written description, even when a specification describes very specific embodiments of the invention . . . unless the specification makes clear that the patentee intends for the claims and the embodiments in the specification to be strictly coextensive." *See Briggs & Stratton Corp. v. Kohler Co.*, 398 F. Supp. 2d 925, 964 (W.D. Wis. 2005) (Crabb, J.) (citing *JVW Enters. v. Interact Accessories, Inc.*, 424 F.3d 1324 (Fed. Cir. 2005)) (internal quotations omitted).[6] "[T]he claims of the patent will not be read restrictively unless the patentee has demonstrated a clear intention to limit the claim scope using words or expressions of manifest exclusion or restriction." *Liebel-Flarsheim*, 358 F.3d at 906. Contrary to an expression of a "clear" intent for Claim 1 to be "strictly coextensive" with the preferred embodiment or any "words or expressions of manifest exclusion or restriction," Claim 1 is specifically and unambiguously directed to an invention utilizing Tiers 1 and 2; dependent Claim 5 utilizes all three Tiers (introducing the synchronization of load/store pairs). Thus, Intel's attempt to import the "load/store pair" limitation into the definition of "data speculation circuit" of Claim 1 should be rejected.

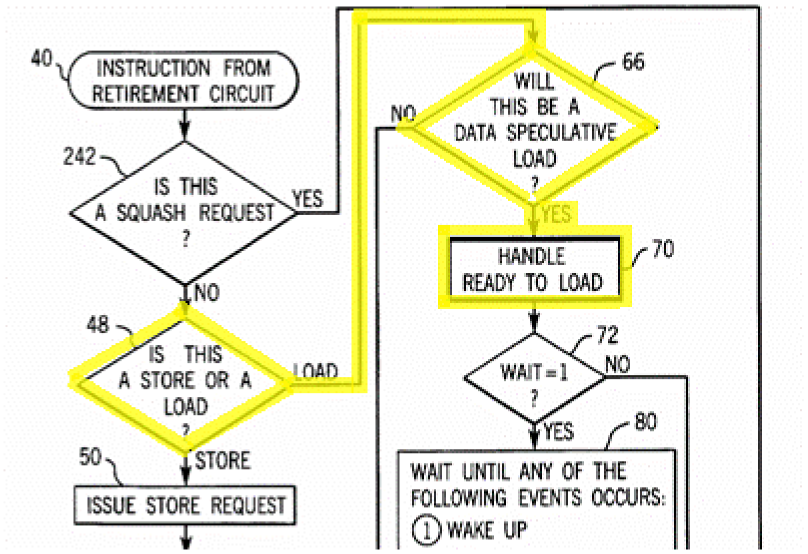
B. Claim 1 Delays A LOAD Instruction Based Only On The Prediction

As pointed out in the Introduction, the crux of the dispute is whether the '752 invention delays a LOAD instruction with regard to a particular STORE instruction, as Intel argues, or whether the decision to delay a LOAD is based only on the associated prediction. Viewed from the perspective of the disclosed prediction table, the dispute is whether the "data speculation circuit," when making the decision to delay the LOAD, refers to both the first column (LOAD) and the second column (STORE) of the table or whether the circuit can make that decision without reference to the second column. The '752 patent discloses that the "data speculation

[6] Statements from the Summary of the Invention are not per se statements of "manifest exclusion or restriction." *See MBO Labs., Inc. v. Becton, Dickinson & Co.*, 474 F.3d 1323, 1330 (Fed. Cir. 2007) ("The summary is of course not wholly dispositive.").

circuit" practices Claim 1 (*second tier*) based on only the identity of the LOAD instruction (without reference to the second column, which contains the paired STORE instruction) and the associated prediction, which is held in the third column of the prediction table. In particular, Figure 3 and columns 9 and 10 provide this disclosure.

Prior to issuing a data instruction, the "data speculation circuit" checks (block 48 of Figure 3) whether it is handling a LOAD or a STORE instruction. Umberger Decl. Ex. 1, Col. 9, ll. 53-55.

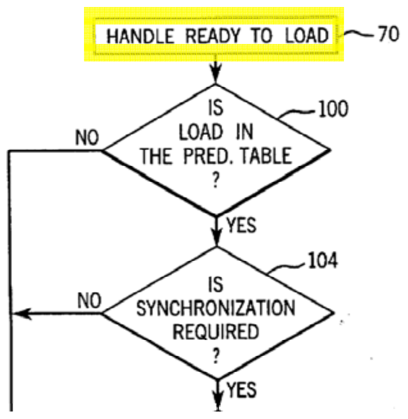

40
INSTRUCTION FROM RETIREMENT CIRCUIT
242
IS THIS A SQUASH REQUEST ?
YES
NO
48
IS THIS A STORE OR A LOAD ?
LOAD
STORE
50
ISSUE STORE REQUEST
66
WILL THIS BE A DATA SPECULATIVE LOAD ?
NO
YES
HANDLE READY TO LOAD
70
72
WAIT=1 ?
NO
YES
80
WAIT UNTIL ANY OF THE FOLLOWING EVENTS OCCURS:
1 WAKE UP

FROM FIG. 3 (DATA SPECULATION CIRCUIT)

If the circuit is handling a LOAD instruction, then it moves to block 66 to verify whether the LOAD instruction is a data speculative LOAD instruction, i.e., "whether there are prior STORE instructions upon which . . . [the LOAD] might depend." Umberger Decl. Ex. 1, Col. 10, ll. 10-11. Notably, the circuit does not check for the paired STORE instruction. If the LOAD instruction is data speculative, then the "data speculation circuit" consults with the predictor circuit (in block 70) to decide whether to speculatively issue the LOAD or to delay it.

Id. at Col. 10, ll. 11-17. The "data speculation circuit" communicates to the predictor circuit through the HANDLE READY TO LOAD signal. *Id.*

Figure 4 describes an embodiment of the "predictor" circuit, which receives the HANDLE READY TO LOAD (70) signal.[7] In block 100, it then checks whether the LOAD instruction is in the prediction table. *Id.* at Col. 11, ll. 15-19. The "predictor" does not check whether the LOAD/STORE pair is in the prediction table. If the LOAD is in the prediction table, then in block 104, the "predictor" checks whether the prediction indicates that synchronization is required. *Id.* at Col. 11, ll. 33-41. If the likelihood of mis-speculation is high, then the LOAD and STORE must synchronize. *Id.* at Col. 10, ll. 18-24 and 35-37; Col. 11, ll. 33-41. In other words, the LOAD instruction must be delayed.



FROM FIG. 4 (PREDICTOR CIRCUIT)

Once the predictor completes block 104, the decision whether to delay or execute a LOAD instruction has been made. This also happens to be the scope of Claim 1 of the '752 patent. Notably, the decision to delay or not to delay the LOAD is made only through reference

[7] The data speculation circuit disclosed in the '752 patent provides other signals to the predictor circuit. Umberger Decl. Ex. 1, Col. 8, ll. 8-34.

to the prediction for a particular LOAD in the prediction table. That decision is made independently of the STORE instruction maintained in the prediction table.[8] Thus, Intel is mistaken in arguing that the ’752 invention operates only on the “load/store pair” basis.

Because significant aspects of the ’752 patent operate without implicating the STORE instruction in the prediction table, Intel’s efforts to limit Claim 1 to unrelated features of the ’752 invention are misdirected. A claim is not limited only to the embodiments or parts thereof disclosed in the specification. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1323 (Fed. Cir. 2005) (“[A]lthough the specification often describes very specific embodiments of the invention, we have repeatedly warned against confining the claims to those embodiments.”). Indeed, it is the claims, not the specification, that “define the invention to which the patentee is entitled the right to exclude.” *Innova/Pure Water, Inc. v. Safari Water Filtration Sys.*, 381 F.3d 1111, 1115 (Fed. Cir. 2004); *see also John Mezzalingua Assocs. v. Arris Int’l, Inc.*, 298 F. Supp. 2d 813, 817 (W.D. Wis. 2003) (Crabb, C.J.) (“[T]he language of the claim defines the scope of the protected invention.”); 35 U.S.C. § 112. Moreover, although the written description serves to aid in the Court’s understanding of the claims, one may not import limitations from the specification into the claims where, as here, the claim does not include that limitation. *See, e.g.*, *Primos, Inc. v. Hunter’s Specialties*, 451 F.3d 841, 848 (Fed. Cir. 2006); *Superguide*, 358 F.3d at 875.

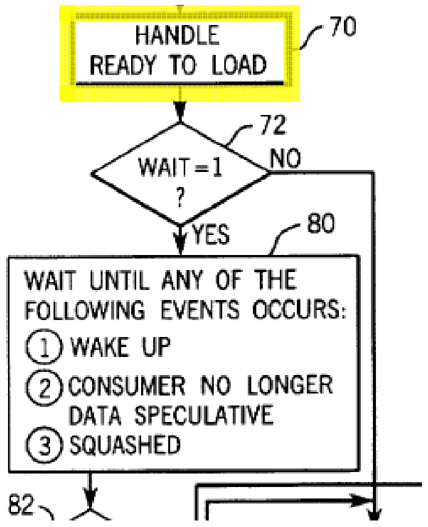
C. The Load Instruction Is Delayed Independent Of The Paired Store Instruction

The error in Intel’s argument that a “particular LOAD” is delayed when a “particular STORE” appears is further revealed by reviewing how the ’752 patent handles delayed LOAD

[8] Only after the decision to delay the LOAD instruction has been made, the predictor circuit checks the synchronization table. Umberger Decl. Ex. 1, Col. 11, ll. 38-41. The description of how the synchronization table is employed by the dependent claims of the ’752 patent is provided in Section I.G. below.

instructions. In deciding when to release a delayed LOAD, the ’752 patent addresses situations when the “particular STORE” appears and when it does **not** appear. Both situations are contemplated because the decision to delay the LOAD was independent of the occurrence of the “particular STORE” instruction.

Reverting to Figure 3 (which describes the “data speculation circuit”), the circuit checks (in block 72) whether the LOAD instruction should be delayed. Umberger Decl. Ex. 1, Col. 10, ll. 18-25. In the disclosed embodiment, if the value of the Wait flag is 1, then the LOAD is delayed. *Id.* at Col. 10, ll. 35-38. The circuit transitions to block 80, and, the delayed LOAD instruction waits until one of the following events occur:[9]


HANDLE READY TO LOAD
70
72
WAIT=1 ?
NO
YES
80
WAIT UNTIL ANY OF THE FOLLOWING EVENTS OCCURS:
1 WAKE UP
2 CONSUMER NO LONGER DATA SPECULATIVE
3 SQUASHED
82

FROM FIG. 3 (DATA SPECULATION CIRCUIT)

A. Consumer No Longer Data Speculative: The LOAD is delayed until any conflicts between the delayed LOAD and **all** the pending STORES have been avoided. Umberger Decl. Ex. 1, Col.

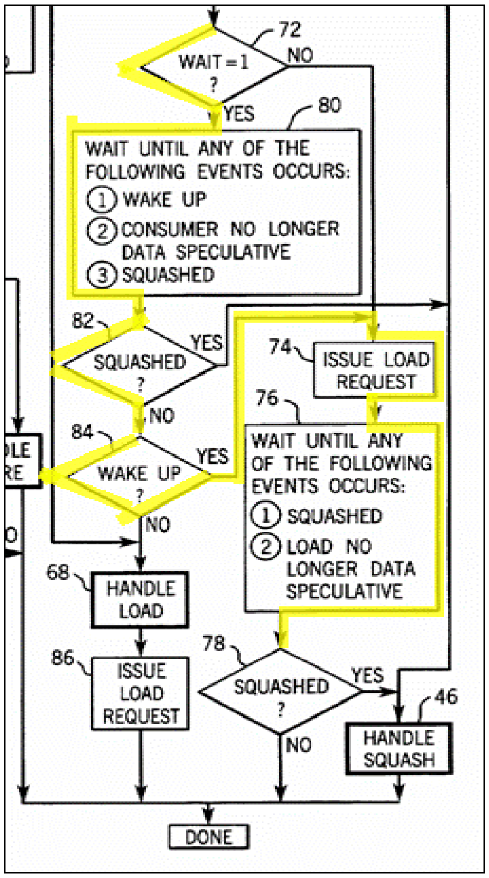
[9] The LOAD instruction can also be squashed. Umberger Decl. Ex. 1, Col. 10, ll. 37-44 (“the data speculation circuit 30 waits . . . for a squash signal indicating that the instruction should be squashed as a result of a later occurring control or data dependency mis-speculation”). This aspect of the invention is not relevant to the dispute about the role of “load/store pair” in the ’752 patent.

10, ll. 38-43 ("the data speculation circuit 30 waits for . . . a signal indicating that the LOAD instruction is no longer speculative because the earlier STORE instructions did not write to its memory location").

In this scenario, the STORE instruction that is paired with the LOAD instruction in the prediction table does **not** execute. The '752 patent describes this situation because the LOAD instruction was delayed irrespective of whether the paired STORE appears or not. By limiting the '752 invention to operate only on the identified "load/store pair" basis, Intel's construction would render this aspect of the invention superfluous. *See Verizon Servs. Corp. v. Vonage Holdings Corp.*, 503 F.3d 1295, 1305 (Fed. Cir. 2007) ("We normally do not interpret claim terms in a way that excludes disclosed examples in the specification.").

B. Wake up: This event corresponds to the extension (i.e., the *third tier*) that is utilized only after a LOAD instruction has been delayed to ascertain when the delayed LOAD should be released. According to the '752 patent, a good time to release the LOAD instruction is when the STORE that is "highly likely" to conflict with the LOAD executes because it "highly likely" that the dependency has been resolved. Umberger Decl. Ex. 1, Col. 12, ll. 50-58. Thus, when the paired STORE occurs, it provides data that the LOAD was waiting for (wakes up the LOAD) and the LOAD is released. Umberger Decl. Ex. 1, Col. 10, ll. 37-39 ("the data speculation circuit 30 waits for a wakeup signal indicating that the dependent STORE instruction has been executed").

By arguing that the purpose of the '752 invention is to "prevent . . . mis-speculation by a load instruction when it appears with a particular store instruction," Intel seeks to limit the entire '752 invention to the *third tier* of the invention. Intel Markman Br. at 32. It is evident that Intel cannot provide any coherent justification for doing so. In any event, Intel's understanding of the *third tier* is incorrect as well. Intel seems to argue that when the "particular STORE" has appeared, mis-speculation by the "particular LOAD" has been prevented. Intel is incorrect again because even though the delayed LOAD is released when the paired STORE occurs, the LOAD is executed in a data speculative manner because the LOAD can still conflict with another pending STORE instruction.



Continuing with **Figure 3**, subsequent to the occurrence of events in block 80, the woken up LOAD instructions transition through blocks 82 and 84 to block 74, where the woken up LOAD is issued in a data speculative manner. Umberger Decl. Ex. 1, Col. 10, ll. 45-51. Because this LOAD can still depend upon any of the pending STORE instructions, as shown in block 76, the woken up LOAD waits until the LOAD is no longer data speculative, i.e., the woken up LOAD does not conflict with any of the pending STORE instructions.[10] Umberger Decl. Ex. 1, Col. 10, ll. 26-31 ("the data speculation circuit 30 waits for that particular instruction . . . for an indication that it is no longer data speculative, that is any previous STORE instructions were for different memory addresses").[11]

In summary, even the preferred embodiment of the '752 invention does not conform to Intel's view of the invention. Though the disclosed prediction table maintains a LOAD and a STORE instruction pair, the *second tier* (which is Claim 1) operates independently of the STORE instruction. Because practicing Claim 1 does **not** necessitate the STORE instruction, it would be an error to import the "load/store pair" limitation into that claim.

Intel cites several cases in support of its argument that the Court should limit the scope of the claims to the patentees' description of the invention in the specification. Intel Markman Br. at 21, 30 (citing *Inpro II Licensing S.A.R.L. v. T-Mobile USA, Inc.*, 450 F.3d 1350 (Fed. Cir. 2006), *Alloc, Inc. v. ITC*, 342 F.3d 1361 (Fed. Cir. 2003), *Honeywell Int'l, Inc. v. ITT Indus.*, 452 F.3d 1312, 1318-19 (Fed. Cir. 2006), and *Microsoft Corp. v. Multi-Tech Sys.*, 357 F.3d 1340, 1348-49 (Fed. Cir. 2004)). Even in those cases, however, the Federal Circuit noted that a limitation not explicit in the claims may only be found where the intrinsic evidence makes clear, i.e., creates the "inescapable conclusion," that the limitation must be included in every

[10] Of course, as block 76 indicates, the LOAD may get squashed. Umberger Decl. Ex. 1, Col. 10, ll. 26-28 ("Next at process block 76, the data speculation circuit 30 waits for that particular instruction, either to be squashed indicating that it has been erroneously speculated . . . .").

[11] Reverting back to box 72 of Figure 3, it is possible that the value of the Wait flag was not 1. In that event, in block 74, the LOAD instruction is speculatively executed. Subsequently, in block 76, the LOAD instruction waits until it is squashed or until it is ascertained that the LOAD will not conflict with any of the pending STORE instructions. Umberger Decl. Ex. 1, Col. 10, ll. 26-34.

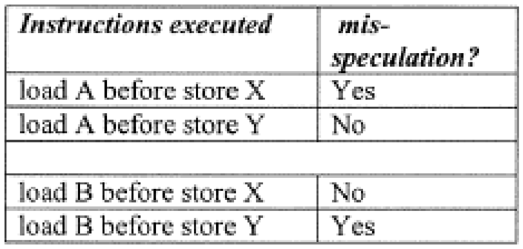
embodiment. *See Inpro*, 450 F.3d at 1355 (finding that limitation included a particular feature where the "specification characterizes the [feature] as a 'very important feature' of the invention" and noting that "*[w]here the specification makes clear that the invention does not include a particular feature*, that feature is deemed to be outside the reach of the claims of the patent . . .") (emphasis added; internal quotation marks omitted); *Alloc, Inc.*, 342 F.3d at 1370-71 (finding limitation where "the very character of the invention requires the limitation be a part of every embodiment" and the "specification read as a whole leads to the inescapable conclusion that the claimed invention must include [the limitation] in every embodiment"; further noting prior case where it refused to apply a limitation from the specification where the specification "did not mandate that the claimed invention include a particular feature"); *Honeywell*, 452 F.3d at 1318-19 (limiting "fuel injection system component" to a "fuel filter" where "the written description uses language that leads us to the conclusion that a fuel filter is the only 'fuel injection system component' that the claims cover," and "[n]o other fuel injection system component with the claimed limitations is disclosed or suggested"); *Microsoft*, 357 F.3d at 1348 ("clear statements in the specification that the invention . . . is directed to communications 'over a standard telephone line,' . . . lead[] to the 'inescapable conclusion' that" the claims are so limited).[12]

[12] Intel likewise suggests that the claims can by properly limited to the scope of a single embodiment disclosed in the specification, and cites several cases in support of this proposition. Intel Markman Br. at 21, 32 (citing *Nystrom v. Trex Co.*, 424 F.3d 1136 (Fed. Cir. 2005), *Watts v. XL Sys.*, 232 F.3d 877 (Fed. Cir. 2000), *Curtiss-Wright Flow Control Corp. v. Velan, Inc.*, 438 F.3d 1374 (Fed. Cir. 2006), and *Honeywell Int'l, Inc. v. Universal Avionics Sys. Corp.*, 493 F.3d 1358 (Fed. Cir. 2007)). Again in those cases, however, the Federal Circuit adopted limitations only where the specification clearly limited the invention and there was no intrinsic evidence to the contrary. *See Nystrom*, 424 F.3d at 1145 (a "board" must be made of wood where "there was *nothing in the intrinsic record* to support the conclusion that a skilled artisan would have construed the term 'board' more broadly than a piece of construction material made from wood cut from a log") (emphasis added); *Watts*, 232 F.3d at 883 (structure limited to one utilizing

(Footnote continued)

D. Intel's Example Of How The '752 Invention Operates Is Incorrect

An example described in Intel's brief further evidences Intel's misunderstanding of the '752 invention. Specifically, Intel provides the following example (Intel Markman Br. at 33) to show that the '752 invention delays a LOAD only when the paired STORE appears:[13]



Instructions executed	***mis-speculation?***
load A before store X	Yes
load A before store Y	No
load B before store X	No
load B before store Y	Yes

Intel argues that in the '752 patent, "data speculation is not being prevented for the load instruction generally, but for the load instruction when it appears with the store instruction." Intel Markman Br. at 32. As a result, with respect to the above example, Intel states:

> "The whole purpose of the invention is to disable speculative execution of the **<u>pair</u>** A-X, not A or X individually."

Intel Markman Br. at 33 (emphasis in original). In other words, Intel argues that the '752 invention delays "load A" only when "store X" occurs and does not delay "load A" when "store Y" occurs. Declaration of William J. Dally in Support of Plaintiff's Response to Intel Corporation's Opening Markman Brief (hereinafter "Dally Supp. Decl.") ¶¶ 10-11. Intel is

---

misaligned taper angles where "the specification actually limits the invention to structures that utilize misaligned taper angles" and the specification does not "disclose[] an embodiment without misaligned taper angles"); *Curtiss-Wright*, 438 F.3d at 1379 (limiting claim term where "the specification . . . consistently, and without exception," describes the term in a particular manner, and where not limiting the term in reference to the specification "renders that limitation nearly meaningless"); *Honeywell*, 493 F.3d at 1362 (limiting claim term where the "specification . . . clearly communicates the meaning the patentees have assigned to the term" and "discloses no other form . . .").

[13] Intel's expert has submitted a voluminous declaration which is extensively relied upon throughout Intel's opening brief. This section of Intel's brief notably lacks the support of its expert.

incorrect. The ’752 patent discloses that a LOAD instruction is delayed when its prediction so dictates irrespective of whether its paired STORE occurs. Dally Supp. Decl. ¶ 12. In the above example, “load A” will be delayed if its associated prediction indicates that data speculative execution of “load A” is likely to result in a mis-speculation. *Id.*

In Intel’s example, “only two scenarios (“pairs”) cause an error: executing ‘A’ ahead of ‘X’; or ‘B’ ahead of ‘Y’.” Intel Markman Br. at 33. Thus, once the processor identifies the two problem “pairs” or scenarios, it will create two entries in the prediction table as shown below:

PREDICTION TABLE		
:	:	:
Load A	**Store X**	**Prediction A-X**
:	:	:
Load B	**Store Y**	**Prediction B-Y**
:	:	:

The prediction table will be updated to generate a “Prediction A-X” associated with “Load A” and a “Prediction B-Y” associated with “Load B” as shown above. Now, when deciding whether to delay “Load A,” the data speculation circuit will consult the predictor (block 70 in Figure 3) to decide whether to delay or execute Load A. The decision is based on the prediction associated with Load A (i.e., Prediction A-X) without requiring a check whether the paired Store X occurs or not (boxes 100 and 104 in Figure 4). If Prediction A-X requires, Load A is delayed, otherwise not. If Load A is delayed, then it waits for one of several events to occurs (block 80 in Figure 3). If Store X executes (i.e., Load A is woken up), the Load A is released. However, as shown in block 80, Load A may be delayed until it can be confirmed that it does not conflict with any of the pending STORE instructions. This scenario occurs only because Store X did not execute and would not occur if Load A is delayed, as Intel contends, when Store X appears.[14] It is possible

[14] It is also possible that Load A may be squashed.

that "Store Y" could have been one of these pending STORE instructions. In that event, contrary to Intel's assertion, Load A would be delayed when Store Y occurs. Thus, the inescapable conclusion is that Intel's assumption of how the '752 patent operates is incorrect.

Finally, if the data speculation circuit is limited to detecting and preventing mis-speculation between identified LOAD/STORE pairs, as Intel suggests, even the example suggested by Intel will result in faulty program execution. It is possible that Load A may depend upon Store Y as program behavior changes and may result in a mis-speculation, which would go undetected. Dally Supp. Decl. ¶ 10. Fortunately, the '752 invention (operating in accordance with WARF's proposal) would execute Intel's example correctly because it would detect all mis-speculations, including the one mentioned above. *Id.* at ¶ 11.

In summary, Load A is delayed when Store X occurs and does not occur. In fact, Load A may be even delayed when Store Y occurs. The purpose of the invention is to disable speculative execution of the LOAD instructions individually based on the prediction associated with those instructions.

E. WARF's Construction Mirrors The Claim And The Specification

WARF's definition of "data speculation circuit" places three requirements on the circuit: (1) detects data dependence between LOAD and STORE instructions, (2) tracks execution of such instructions, and (3) detects data mis-speculation by LOAD instructions. WARF's definition is correct because, as shown below, it begins with the requirements specified in Claim 1 and reaches out to the '752 specification to convey the full import of those requirements.

The preamble of Claim 1 requires a:

> **data speculation circuit** for detecting data dependence between instructions and detecting a mis-speculation where a data consuming instruction dependent for its data on a data producing instruction of earlier program order, is in fact executed before the data producing instruction

Umberger Decl. Ex. 1, Claim 1 (emphasis added). In other words, the preamble requires the "data speculation circuit" to (1) detect data dependence between instructions and (2) to detect data mis-speculation. The '752 patent mirrors those requirements. *See* Umberger Decl. Ex. 1, Col. 2, l. 64 – Col. 3, l. 4 ("A data speculation circuit may exist to **detect data dependencies** and report any mis-speculation to a retirement circuit. . . . The retirement circuitry also resolves **mis-speculation detected** by the data speculation circuit . . . .") (emphasis added).

WARF's construction reflects the specification, which explains that "detecting data dependence between instruction" implies that the "data speculation circuit" must track the execution of all LOAD and STORE instructions that are being executed by the processor and not just the identified LOAD/STORE pairs:

> The data speculation circuit 30 receives signals . . . that notify it of the program order of any instructions . . . that will access memory. The data speculation circuit is responsible of **keeping track of order of the memory operations** as they are performed . . . so that it can detect any mis-speculations."

Umberger Decl. Ex. 1, Col. 7, ll. 1-7 (emphasis added).

Intel does not dispute this. Intel Markman Br. at 34. "Memory operations" refer to LOAD and STORE operations and not LOAD/STORE pairs. Similarly, for every STORE instruction that it handles, "the data speculation circuit 30 checks other **concurrent LOAD instructions** to see if they have been prematurely executed and thus whether there has been a mis-speculation." Umberger Decl. Ex. 1, Col. 9, ll. 61-64 (emphasis added). And prior to retiring any STORE instruction (not just a STORE instruction in a LOAD/STORE pair), the '752 patent states:

> Whenever **a store instruction** is ready to commit and write its data to a memory address, the data speculation circuit 30, checks to see if any subsequent [**load instructions**] in the instruction window . . . have accessed the same memory address, and if so instructs the allocation circuit 20 and retirement circuit 26 that these **load instructions** are to be squashed and re-allocated by the allocation circuit 20 at a later time.

Umberger Decl. Ex. 1, Col. 7, ll. 41-48 (emphasis added).

Thus, the specification consistently mentions that the "data speculation circuit" detects data mis-speculation between LOAD and STORE instructions. Dally Supp. Decl. ¶¶ 5-7.

The last portion of WARF's definition clarifies that data mis-speculation arises from speculative execution of LOAD instructions because only the LOAD instructions, and not STORE instructions, are speculatively executed.[15] As a result, in the context of detecting data mis-speculation, the '752 patent always refers to LOAD instructions being erroneously executed.

Intel states that the parties agree that the second half of the construction concerns detecting mis-speculation by a load instruction, but this is an incorrect characterization of WARF's position. The second half of the construction should at least concern detecting **data** mis-speculation by a load instruction. Addition of "data" is necessary to define the proper scope of the term. Intel's term would include both *data* and *control* mis-speculation.

Finally, Intel appends "in fact executed" to its definition. To lend an aura of legitimacy, Intel asserts that this is mandated by the claim language. Those assertions ring hollow because Intel inexplicably omits significant portions of the preamble. In fact, to faithfully mirror the preamble, Intel would have to add the underlined portion:

> mis-speculation by a load instruction that is in fact executed [before the STORE instruction, where the LOAD instruction depends on its data on the STORE instruction]

[15] Intel agrees. Prior to the filing of the '752 patent, Intel stated in one of its patents (U.S. Patent No. 5,721,857 to Glew et al.) that the STORE instructions are never speculatively executed. *See* Umberger Resp. Decl. Ex. 4, Col. 13, ll. 7-8 ("Stores are never performed speculatively since a memory write is not reversible.").

Notably, WARF's definition captures this through its definition of "mis-speculation" which is incorporated in the definition of "data speculation circuit." Intel's definition of "mis-speculation" also omits these relevant portions of Claim 1.

F. The Term "Pair" Does Not Appear In Claim 1

Finally, because the term "pair" does not even occur in Claim 1, Intel is forced to distort the plain meaning of the claim in order to find the elusive "pair" in that claim. Intel Markman Br. at 28-29. None of its arguments have any merit.

First, Intel argues that the preamble identifies a LOAD/STORE pair. That is incorrect. The preamble simply defines the event that constitutes data mis-speculation. The preamble requires a data speculation circuit to detect mis-speculations, which it defines as when <u>**a**</u> LOAD instruction, dependent for its data on <u>**a**</u> STORE instruction, is in fact executed before <u>**the**</u> STORE instruction. A LOAD instruction can depend upon several STORE instructions, and thus, the preamble refers to <u>**a**</u> STORE instruction. Dally Supp. Decl. ¶ 8. "As a general rule," the words "a" or "an" in a patent claim carry the meaning of 'one or more.'" *See, e.g., Baldwin Graphic Sys., Inc. v. Siebert, Inc.*, 512 F.3d 1338, 1342 (Fed. Cir. 2008).

When defining data speculation, the '752 specification also indicates that a LOAD can depend upon one or more STORE instructions:

> Data speculation, for example, might involve reading from memory [i.e., LOAD instruction] to obtain data for a later instruction, even though there are earlier <u>**STORES**</u> to that memory location that have not yet been completed and that may change the value of the memory location.

Umberger Decl. Ex. 1, Col. 2, ll. 36-40 (emphasis added).

The concept that the LOAD can depend upon several STORE instructions is well-known in the art. Dally Supp. Decl. ¶ 20. *See* Declaration of Michelle M. Umberger in Support of

Plaintiff's Response to Defendant Intel Corporation's Opening Markman Brief ("Umberger Resp. Decl.") at Ex. 2.

Data mis-speculation occurs when the LOAD executes before one of the STORE instructions it depends upon. Thus, the preamble appends the article **the** before the second reference to the STORE instruction because it points to **the** STORE instruction that actually conflicted with the LOAD. The conflicted STORE is one of the several possible STORE instructions that can conflict with the LOAD. Each time a LOAD instruction executes, it may not necessarily conflict with the same STORE. The language of the claim does not indicate any pairing of LOAD and STORE instructions. Dally Supp. Decl. ¶ 8. Rather, it defines what constitutes a data mis-speculation.

Second, Intel also argues that the term "mis-speculation indication" in Limitation (a) of Claim 1 refers to the mis-speculation mentioned in the preamble. Intel bases its argument on the use of article "the" before mis-speculation indication in Limitation (a). Intel Markman Br. at 28-29. Intel argues that mis-speculation mentioned in the preamble provides the antecedent basis for the term "the mis-speculation indication" in Limitation (a). Intel's interpretation of the claim language is incorrect. Limitation (a) refers to mis-speculation indication twice:

> a predictor receiving **a** **mis-speculation indication** from the data speculation circuit to produce a prediction associated with the particular data consuming instruction and based on **the** **mis-speculation indication**;

Umberger Decl. Ex. 1, Claim 1.

The second reference, which is preceded by the article "the" and forms the basis of Intel's argument, merely refers to the first occurrence of the mis-speculation indication, which is preceded by article "a". Thus, Intel's argument that the presence of the article "the" somehow connects "mis-speculation indication" of Limitation (a) to the LOAD and STORE instructions in the preamble is incorrect. The only explicit reference to an instruction in the preamble is the LOAD instruction ("the particular data consuming instruction").[16]

G. The Synchronization Table Is Utilized By The Dependent Claims Of The '752 Patent

As mentioned in Section I.A., the synchronization table and the STORE instructions in the prediction table are implicated in the *third tier* of the '752 invention only after a LOAD instruction has been delayed, i.e., when the invention is operating outside the ambit of Claim 1. The operation of that tier pertains to ascertaining when the delayed LOAD instruction should be released and is captured in the dependent claims of the '752 patent. "As [the Federal Circuit] has frequently stated, the presence of a dependent claim that adds a particular limitation raises a presumption that the limitation in question is not found in the independent claim." *Liebel-Flarsheim*, 358 F.3d at 910; *Phillips*, 415 F.3d at 1314-15 ("presumption that the limitation in question [present in dependent claim] is not present in the independent claim") (citing *Laitram Corp. v. Rexnord, Inc.*, 939 F.2d 1533, 1538 (Fed. Cir. 1991)). Because Intel makes frequent references to the synchronization table in an effort to conflate the entire '752 invention with the *third tier*, WARF discusses below how the synchronization table and the STORE instructions are utilized when operating that tier of the invention. WARF first discusses how LOAD and STORE instructions are handled by the '752 patent after a LOAD instruction has been delayed:

<u>Handling a LOAD instruction</u>:

In Figure 4, after the "predictor" has ascertained that the LOAD instruction should be delayed, it takes steps to enable the third tier of the '752 invention. Umberger Decl. Ex. 1, Col.

[16] On page 29 of its opening brief Intel cites to Col. 12, ll. 65-67 of the '752 patent to argue that the invention should be limited to "load/store pair." However, the cited text merely shows how the patent updates the prediction; that procedure is discussed in Section III.B.

11, ll. 38-41. After the decision to delay the LOAD has been taken, the predictor then checks whether the LOAD/STORE pair exists in the synchronization table. *Id.*

If the LOAD/STORE pair is not in the synchronization table, then it implies that the paired STORE instruction that provides data to the LOAD has not yet executed. *Id.* at Col. 11, 48-63. The predictor creates an entry in the synchronization table and places the LOAD and STORE instructions in that entry as depicted below in Figure 6 from the patent. *Id.* The synchronization flag 112 is set to zero to indicate that the STORE has not yet been executed. *Id.* It is important to note that dependent Claim 5 (a claim that depends from Claim 1 that refers to a synchronization table and a flag) claims that a LOAD is delayed when a flag indicates that the paired STORE instruction has not been executed.

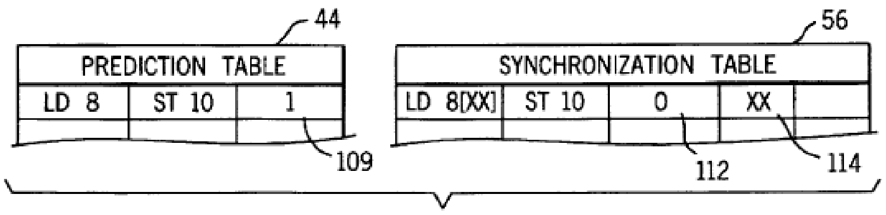


FIG. 6

PREDICTION AND SYNCHRONIZATION TABLES

When the corresponding STORE instruction executes, the STORE will utilize the prediction table to obtain the identity of its paired LOAD instruction, i.e., the LOAD that was used to create the entry in the synchronization table. Umberger Decl. Ex. 1, Col. 12, ll. 23-31. It will then check the synchronization table and find the LOAD/STORE pair in the table. And, if the synchronization flag will be set to zero, the STORE will realize that the LOAD is still waiting. *Id.* at Col. 12, ll. 50-58. The STORE provides the data to the LOAD, the LOAD is released and the synchronization table entry is deleted. *Id.*

If the LOAD/STORE pair is in the synchronization table, then the LOAD checks whether the STORE has executed, i.e., whether the synchronization flag is set to one. If so, then the LOAD can be issued right away and entry in the synchronization table is deleted. *Id.* at Col. 11, l. 58 – Col. 12, l. 19. Dependent Claim 7 related to this aspect of the ’752 patent. If the flag is still zero indicating that the STORE has not yet executed, the LOAD is speculatively executed.

Handling a STORE instruction:

When handling a STORE instruction, the processor similarly takes steps to ensure that the third tier of the invention works correctly. Figure 7 depicts how the STORE instruction is handled by the “predictor” circuit. In blocks 201 and 202 of Figure 7, the predictor checks whether the STORE is in the prediction table and, if so, whether there is a LOAD that is predicted to conflict with the STORE. Umberger Decl. Ex. 1, Col. 12, ll. 23-31. If a LOAD exists, the predictor checks whether the synchronization table contains an entry for the LOAD/STORE pair. *Id.*

The lack of an entry in the table indicates that the corresponding LOAD has not yet occurred. *Id.* at Col. 12, ll. 31-41. In that event, the predictor allocates an entry in the synchronization table and places the LOAD and STORE instructions and depicted in Figure 8 below. It also sets the synchronization flag 112 to one to indicate that the STORE has occurred.

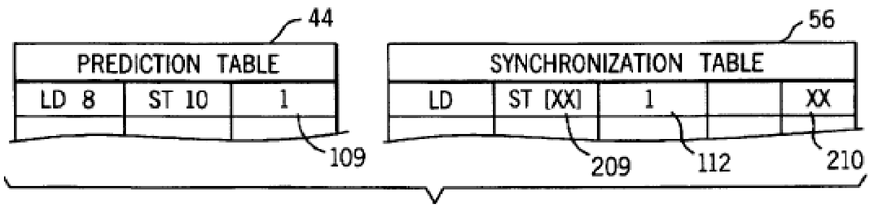


FIG. 8

As discussed above, when the predictor handles the LOAD instruction, it will find the synchronization flag to be one and the LOAD will be released. *Id.* at Col. 11, l. 58 – Col. 12, l. 9.

If the LOAD/STORE entry exists in the synchronization table, the predictor checks the value of the synchronization flag. If the synchronization flag is one, then the program updates the table with the new values. *Id.* at Col. 12, ll. 42-49. If the flag is still zero (indicating that the LOAD is waiting for the STORE), the LOAD is released through a Wake up signal to the LOAD. *Id.* at Col. 12, ll. 42-49.

H. WARF's Construction Is Based On The Correct Interpretation of The '752 Invention

In summary, the preamble defines "data speculation circuit" and narrows "mis-speculation" to only data mis-speculation. It does not limit the data speculation circuit to be limited to detecting and preventing mis-speculations between identified pairs of LOAD and STORE instructions. The data speculation circuit tracks execution of LOAD and STORE instructions and detects data mis-speculations. WARF's proposal reflects that. The prediction table maintains LOAD and STORE pairs in order to decide when to best release a delayed LOAD instruction. That occurs in the *third tier*. Intel incorrectly conflates the *third tier* with the entire '752 invention and thus, its proposed construction should be rejected.

II. "A PREDICTION THRESHOLD DETECTOR PREVENTING DATA SPECULATION FOR INSTRUCTIONS HAVING A PREDICTION WITHIN A PREDETERMINED RANGE"

The dispute regarding Limitation (b) of Claim 1 is the same – whether the LOAD instruction is delayed only when paired the STORE occurs or whether the LOAD instruction is delayed independent of the occurrence of the paired STORE instruction.

Claim Term	WARF's Proposal	Intel's Proposal
a prediction threshold detector preventing data speculation for instructions having a prediction within a predetermined range.	a circuit that prevents data speculation for a particular LOAD instruction when the prediction associated with said LOAD instruction is in a predetermined range.	a circuit that prevents data speculation of a load/store pair if the prediction value for the pair is within a predetermined series of multiple values.

As demonstrated above, Intel's efforts to import the "load/store pair" limitation into the claim should be ignored because they are based on a flawed understanding of the '752 invention. The plain language of Claim 1 supports WARF's construction. Limitation (a) refers to a prediction associated with a particular LOAD instruction. Limitation (b) refers to instructions having a prediction. It naturally follows that Limitation (b) is referring to LOAD instructions. The operation of the "data speculation decision circuit" also supports this construction: when the circuit is handling a LOAD instruction, it obtains from the predictor a prediction associated with the LOAD instruction. The prediction threshold detector decides to delay the LOAD instruction if the prediction associated with that LOAD is within a predetermined range.

Intel's proposed definition is incorrect because it is based on a flawed reading of the '752 patent. WARF's definition should be adopted because it is rooted in the correct interpretation of the '752 invention.

III. "PREDICTION"

Intel further attempts to introduce the notion of "LOAD/STORE pairs" into the construction of the broad term "prediction," contrary to the express language and the context in which that term appears in Limitation (a) of Claim 1 (emphasis added):

> a) a predictor receiving a mis-speculation indication from the data speculation circuit to produce **a <u>prediction</u> associated with the [LOAD instruction] and based on the mis-speculation indication**;

The parties' proposed constructions of "prediction" are:

Claim Term	WARF's Proposal	Intel's Proposal
prediction	a dynamic multi-bit value which indicates the likelihood that the data speculative execution of a LOAD instruction will result in a mis-speculation	a value indicating the likelihood that data speculative execution of a load/store pair will result in a mis-speculation

In its effort to limit "prediction" by importing the "load/store pair" limitation to Claim 1, Intel essentially proffers two arguments: *first*, that the purpose of the '752 invention is to delay a LOAD only when the paired STORE appears, and *second*, that the prediction is simply a count of mis-speculations between a particular LOAD and a particular STORE. Both of these assertions are based on an inaccurate understanding of the '752 patent. Intel's first assertion has been addressed in Section I above and shown to be incorrect. The error of the second is demonstrated in Section III.B. below. Contrary to Intel's argument, "prediction" is not a mis-speculation counter. Because Intel begins with a flawed premise, it arrives at an incorrect construction.

A. Prediction Is The Likelihood Of A Mis-speculation By A LOAD Instruction

It is a bedrock principle of patent law that the claims of the patent define the invention. *See, e.g., Innova*, 381 F.3d at 1115. The claim language is then the best guide to understand the meaning of the term "prediction." *See John Mezzalingua v. Arris*, 298 F. Supp. 2d at 817 ("[T]he language of the claim defines the scope of the protected invention."). Limitation (a) requires "prediction" to be:

- associated with a particular LOAD instruction; and
- based on the historical data mis-speculation pattern of the LOAD instruction.

These two restrictions alone define how the prediction should be calculated. In Limitation (b), the prediction threshold detector delays data speculative execution of the LOAD instruction if the prediction associated with that LOAD instruction indicates that such execution is likely to result in a mis-speculation. WARF's proposed definition of "prediction" as the likelihood that the data speculative execution of a LOAD instruction will result in a mis-speculation is faithful to the claim language. Intel agrees with the overall purpose of "prediction" except it erroneously seeks to insert a "STORE instruction" into the limitations of Claim 1. Notably, neither of the two limitations of Claim 1 refer to a STORE instruction.

WARF's construction of "prediction" as used in Claim 1 reflects the use of that term in the specification, which is "always highly relevant to the claim construction analysis." *See Chamberlain Group v. Lear Corp.*, 516 F.3d 1331, 1335 (Fed. Cir. 2008) (the specification "is the single best guide to the meaning of a disputed term"). Prediction is used as a value associated with a LOAD instruction and as a forecast of the likelihood of a mis-speculation by a LOAD instruction.

In a nutshell, Claim 1 on its face is about delaying LOAD instructions (i.e., "preventing data speculation for instructions") if the associated prediction indicates that their execution is likely to result in a data mis-speculation. Consistent with the claim, with respect to this aspect of the '752 invention, the specification consistently associates prediction with a LOAD instruction and not with a "load/store pair":

- "The instruction synchronization circuit will delay the particular data consuming [i.e., LOAD] instruction only when the **prediction associated with the data consuming [i.e., LOAD] instruction** is within a predetermined range of predictions and when the particular data consuming [i.e., LOAD] instruction is in the prediction table." Umberger Decl. Ex. 1, Col. 4, ll. 41-46 (emphasis added).

- "If there has been a mis-speculation with a given LOAD instruction, a **predictor based on the past history of mis-speculation for that LOAD instruction** is employed to determine whether the instruction should be executed or delayed." Umberger Decl. Ex. 1, Col. 3, l. 67-Col. 4, l. 4 (emphasis added).

The manner in which "prediction" is employed in the '752 patent when issuing a LOAD instruction also demonstrates that the inventors ascribed a broader meaning to "prediction." As discussed in Section I above and in Plaintiff's Opening Brief at pages 26-27, when deciding whether to execute or delay a LOAD instruction that has a history of mis-speculation, the '752 invention obtains from the prediction table *only* the prediction value associated with that instruction. Umberger Decl. Ex. 1, Col. 10, ll. 8-12, 18-24; Col. 11, ll. 27-41. While the invention is operating within the scope of Claim 1, it does not even refer to the STORE instruction in the prediction table and the LOAD instruction is delayed without regard to the appearance of the paired STORE instruction. *Id.*; *see also* Dally Supp. Decl. ¶¶ 16-18. Thus, in Claim 1, prediction is used as the likelihood of mis-speculation of the LOAD instruction with ***any*** pending STORE instruction and not just the paired STORE instruction.

Though one embodiment maintains prediction on a LOAD/STORE pair basis, use of prediction in Claim 1 and in the *second tier* of the '752 invention is more probative of the correct scope of that term. Dally Supp. Decl. ¶ 17. In these situations, prediction is employed to forecast whether data speculative execution of an associated LOAD instruction will result in a mis-speculation. This is precisely what WARF's definition reflects.

B. Intel's Assertion That Prediction Is A Mis-speculation Counter Is Incorrect

Intel's opening brief also argues that construction of "prediction" should contain the term "load/store pair" because prediction is a mis-speculation counter for a pair of LOAD and STORE instructions:

- "In other words, the **prediction is a mis-speculation counter**: a mis-speculation makes the count go higher." Intel Markman Br. at 17 (emphasis added).
- "In other words, the processor of the '752 patent . . . maintains **a mis-speculation counter for a load/store pair** that has mis-speculated, (c) the counter goes up when the speculation by that load/store pair is unsuccessful." *Id.* (emphasis added).

Intel thus contends that, at any instant of time, the value of the prediction will provide the number of times a "load/store pair" has mis-speculated. Intel Markman Br. at 18, 42-43. Intel is incorrect. The prediction does not simply count mis-speculations. It is not even limited to being a counter. Rather, the prediction captures the historical mis-speculation behavior of a LOAD instruction. As Claim 1 requires, prediction is "based on" a mis-speculation indication, which includes both the presence or absence of such an indication.

A review of how the '752 patent handles a mis-speculation and how the prediction is updated ***proves*** that the prediction is not a mis-speculation counter. Figure 9 explains how the predictor handles a mis-speculation between a LOAD and a STORE instruction. *See* Umberger Decl. Ex. 1, Col. 12, l. 60 - Col. 13, l. 48. When a mis-speculation occurs, the prediction table is checked to determine whether the LOAD/STORE pair causing the mis-speculation is in the prediction table. If the pair is found in the table, prediction is updated to indicate that the pair conflicts. Umberger Decl. Ex. 1, block 301 of Figure 9, Col. 12, ll. 64-67.

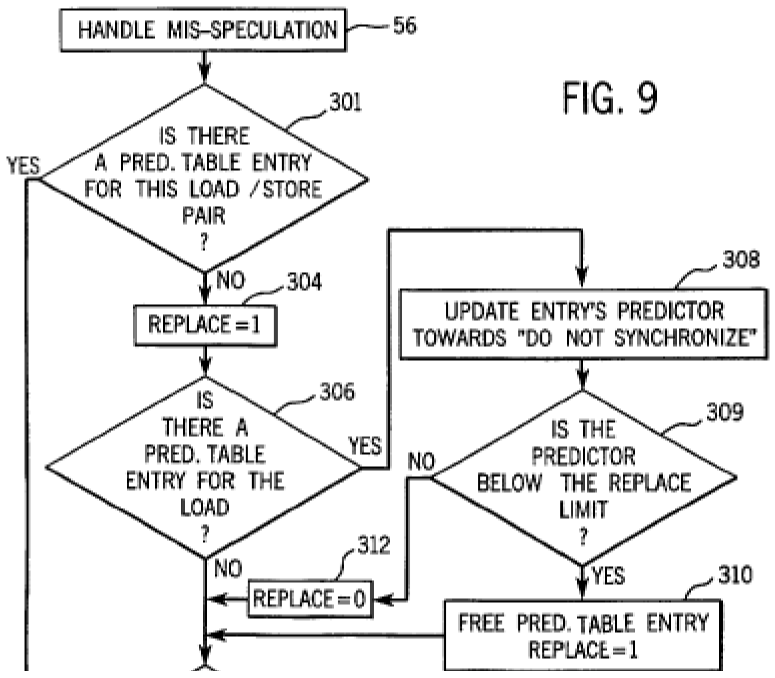

HANDLE MIS-SPECULATION
56
301
IS THERE A PRED. TABLE ENTRY FOR THIS LOAD / STORE PAIR ?
YES
NO
304
REPLACE=1
306
IS THERE A PRED. TABLE ENTRY FOR THE LOAD ?
YES
NO
312
REPLACE=0
308
UPDATE ENTRY'S PREDICTOR TOWARDS "DO NOT SYNCHRONIZE"
309
IS THE PREDICTOR BELOW THE REPLACE LIMIT ?
NO
YES
310
FREE PRED. TABLE ENTRY REPLACE=1

FIG. 9

Intel quotes this portion of the specification to argue that prediction is "a mis-speculation counter for load/store pair that has mis-speculated." Tellingly, Intel neglects to explain what happens in case the LOAD/STORE pair is not found in the prediction table.

Before analyzing the remainder of Figure 9, it is important to review the temporal aspect of LOAD/STORE pairs. The STORE instruction paired with a LOAD instruction in the prediction table is the STORE (e.g., STORE_1) that is **highly likely** to conflict with the given LOAD instruction during *that phase* of program execution. Umberger Decl. Ex. 1, Col. 11, ll. 30-33 ("The higher the prediction . . . the more likelihood of mis-speculation if the instruction of the first column is executed before the instruction of the second column."); Col. 14, ll. 1-3 (prediction is "used to determine the likelihood of dependency between two instructions in the future"); and Col. 3, ll. 54-57. As the program executes, the LOAD instruction may start conflicting with a different STORE instruction (e.g., STORE_2). Dally Supp. Decl. ¶ 17; Umberger Decl. Ex. 1, Col. 13, ll. 7-13. That occurs when the pattern of conflicts between instructions transitions from LOAD/STORE_1 to LOAD/STORE_2. In the disclosed embodiment, when the change in the conflict pattern is confirmed, the prediction table is accordingly updated, i.e., LOAD/STORE_1 pair is replaced by LOAD/STORE_2 pair. Dally Supp. Decl. ¶ 17; Umberger Decl. Ex. 1, Col. 13, ll. 7-17. Using the example of LOAD, STORE_1 and STORE_2 instructions, WARF demonstrates that prediction is **not** a mis-speculation counter:

- When a conflict between LOAD and STORE_1 is detected, the predictor (in block 301 of Figure 9) checks whether the LOAD/STORE_1 pair exists in the prediction table. Because the pair is currently conflicting, an entry for this pair will exist in the table. The prediction for this pair is then updated. Umberger Decl. Ex. 1, Col. 12, l. 64-Col. 13, l. 2.

- As the locality changes, LOAD will start to conflict with STORE_2. When a conflict between LOAD and STORE_2 is detected, the predictor (in block 301) will not find

> the LOAD/STORE_2 instruction pair in the prediction table.[17] However, the predictor does not stop here. *Id.* at Col. 13, ll. 3-6. The predictor (in block 306 of Figure 9) checks whether there is an entry for that LOAD instruction only. The predictor will find the LOAD/STORE_1 entry. And, the prediction value for LOAD/STORE_1 instruction is updated (in block 308).[18] *Id.* at Col. 13, ll. 8-17.

Note that the prediction value of LOAD/STORE 1 is updated based on a conflict between a different instruction pair – the LOAD/STORE 2 instruction pair. Such an event can occur several times. In other words, the prediction for LOAD/STORE_1 pair is **<u>not</u>** counting mis-speculations between the LOAD and STORE_1 instructions. Thus, Intel's assertion that the prediction is a "mis-speculation counter for a load/store pair" is incorrect.[19]

Intel's assertion that "prediction" as used in the '752 patent is merely a counter is also incorrect. Intel Markman Br. at 17, 42-43. While the description of the preferred embodiment makes reference to prediction being incremented or decremented, it also refers to prediction being "updated" (Umberger Decl. Ex. 1, Col. 12, l. 53; Col. 14, l. 1) or the prediction being "moved toward" (*Id.* at Col. 13, ll. 14, 26), which itself implies something more than merely counting up or down. The specification specifically states: "It will be understood that the **prediction 109 may be obtained by methods other than simply incrementing** it in value for

[17] In the preferred embodiment, the prediction table contains only one entry for any LOAD instruction. Umberger Decl. Ex. 1, Col. 13, ll. 38-47.

[18] In the bottom half of Figure 9, the predictor repeats the above steps for the STORE_1 instruction. It locates all table entries containing STORE_1 and updates the associated predictions. Umberger Decl. Ex. 1, Col. 13, ll. 20-30.

[19] The procedure employed by the '752 patent to update the prediction table highlights **another** manner in which the '752 patent employs prediction. In block 309 of Figure 9, the predictor checks the updated prediction for LOAD/STORE_1. If the prediction falls below a "Replace Limit" indicating that the conflict pattern has transitioned from LOAD/STORE_1 to LOAD/STORE_2 pair, the LOAD/STORE_1 entry is replaced by the LOAD/STORE_2 entry in the prediction table. Umberger Decl. Ex. 1, Col. 13, ll. 13-17 and ll. 31-36. Thus, the '752 patent uses "prediction" to identify the threshold when an instruction pair should be replaced by another instruction pair in the prediction table due to a change in the conflict pattern.

each speculation as in described herein." Umberger Decl. Ex. 1, Col. 14, ll. 6-9 (emphasis added). And, the specification goes on to detail such methods of obtaining prediction. Umberger Decl. Ex. 1, Col. 14, ll. 9-14.

A review of the dependent claims also implies that "prediction" is not limited to being a counter. Claim 8, which depends from Claim 1, implies that prediction can be obtained by "tallying" or counting mis-speculation indications.[20] Umberger Decl. Ex. 1, Claim 8. Thus, the presumption is that prediction can be calculated in ways other than simply counting mis-speculation indications. *See Phillips*, 415 F.3d at 1314-15 ("Differences among claims can also be a useful guide in understanding the meaning of particular claim terms. For example, the presence of a dependent claim that adds a particular limitation gives rise to a presumption that the limitation in question is not present in the independent claim.") (citations omitted). In light of the disclosure in the specification and the claims, Intel has no basis to argue that the "prediction" is limited to a counter for mis-speculations between a specific LOAD/STORE pair.

Finally, it is important to remember that Claim 1 requires the prediction to be **based** on one or more mis-speculation indications. In the '752 patent, prediction is updated not only when a mis-speculation occurs but also when mis-speculation does not occur or has been avoided. Thus, prediction is based on both the presence and the absence of a mis-speculation indications. Specifically, the '752 patent discloses updating the prediction when the following occur:

- When a delayed LOAD is released: A delayed LOAD is released after it is confirmed that it will not conflict with any of the pending STORE instructions. Thus, based on a lack of mis-speculation, the prediction for that LOAD is updated to indicate that the LOAD is less likely to conflict with another STORE. Umberger Decl. Ex. 1, Col. 13, ll. 50-59.

[20] Claim 8 contains a minor typographical error. In spite of the error, it is clear that Claim 8 indicates that the predictor produces a prediction by tallying the mis-speculation indications for a LOAD instruction.

- When a delayed LOAD that found its paired STORE is issued: To support the *third tier*, before issuing a LOAD, the predictor checks whether the paired STORE has issued. If the paired STORE has been executed, then the LOAD and the STORE did not conflict. Based on this lack of mis-speculation, the prediction is updated to indicate that the LOAD and the STORE are less likely to conflict. Umberger Decl. Ex. 1, Col. 12, ll. 10-19.

- When a STORE finds a pending LOAD: When the predictor handles a STORE, it checks whether a LOAD is waiting for this STORE. If a LOAD is waiting, then the processor just avoided a mis-speculation. The prediction is updated to indicate that the pair is likely to conflict based on the avoided (or a lack of) mis-speculation. Umberger Decl. Ex. 1, Col. 12, ll. 50-53.

In short, prediction measures past behavior for the LOAD instruction through the occurrence or non-occurrence of a mis-speculation. It is not merely a mis-speculation counter, as Intel portrays it to be.

C. The Preferred Embodiment Maintains LOAD/STORE Pairs To Support The Third Tier Of The Invention

To justify importation of the term "load/store pair" into Claim 1, Intel repeatedly alludes to that fact that the preferred embodiment maintains a STORE for every LOAD in the prediction table and the associated prediction. As discussed above in Section I, the preferred embodiment maintains the STORE instruction in the table to enable the *third tier* of the invention. Dally Supp. Decl. ¶ 18. The patent does not state (expressly or by implication) that the structure of the disclosed prediction table is essential to practice **all** tiers of the invention. Nor does it mention anywhere that the disclosed structure is the only manner in which the prediction table should be maintained.

In light of the specification, a person of skill will understand that the prediction table employed in the ’752 patent is only one manner of achieving the purpose of the ’752 invention. The disclosed table maintains only one STORE instruction for each LOAD and updates the table when the LOAD begins to conflict with a different STORE. However, to practice the ’752 invention, one could expand the prediction table to include space to associate two or more STORE instructions with each LOAD. Dally Supp. Decl. ¶ 20. In fact, a similar arrangement was proposed shortly after the ’752 patent was filed by Chrysos and Emer in a paper titled “Memory Dependence Prediction Using Store Sets” that was published in 1998.[21] Umberger Resp. Decl. Ex. 2; Dally Supp. Decl. ¶ 20. Such an arrangement would allow a LOAD to be predicted by tracking a set of STORE instructions that the LOAD has been historically dependent on. Dally Supp. Decl. ¶ 20.

Similarly, to practice only Claim 1 and the *first* and *second tiers* of the invention, a skilled person would also visualize an abbreviated prediction table. Dally Supp. Decl. ¶ 19. Because the identity of the STORE instruction is not required to practice Claim 1 (and the *second tier*), such a table would consist of only two columns – one containing a LOAD instruction and the other containing the associated prediction. *Id.* In fact, even the LOAD instruction could be omitted from the prediction table by sacrificing some accuracy. *Id.*

Thus, it would be an error to limit “prediction” to operate only on a load/store pair based on one embodiment of the ’752 patent. The “prediction” is based on the past behavior of the LOAD instruction and is tied to that LOAD instruction. As long as any method of prediction leverages these two concepts, it stays within the bounds of Claim 1. *See* Umberger Decl. Ex. 1, Col. 14, ll. 28-32 (“The above description has been that of a preferred embodiment of the present invention. It will occur to those that practice the art that many modifications may be made without departing from the spirit and scope of the invention.”).

[21] WARF believes that both the authors of that paper, Mr. George Chrysos and Mr. Joel Emer, are presently employed by Intel.

D. The Threshold Controls The Sensitivity To Prediction

Continuing with its efforts to mis-characterize the ’752 invention, Intel argues that “the patented processor is designed **not** to disable speculation on the first instance of a mis-speculation.” Intel Markman Br. at 42 (emphasis in original). It is revealing that Intel fails to cite to the ’752 patent to support this proposition. The ’752 patent teaches and claims a flexible mechanism that allows the ’752 invention to adapt to a variety of situations, including sensitivity to data mis-speculations. The patent specifically discloses that the prediction can be maintained through “various weighting schemes” that allow the invention “to be less sensitive to the earliest mis-speculations” or be more sensitive to such speculations. Umberger Decl. Ex. 1, Col. 14, ll. 9-11. Similarly, the patent claims and teaches a threshold detector that prevents data speculation based on the value of prediction. Umberger Decl. Ex. 1, Claim 1; Col. 4, ll. 21-23. The threshold can be adjusted to make the invention more or less sensitive to the presence or absence of data mis-speculations. Thus, Intel is mistaken in arguing that the ’752 patent cannot disable data speculation at the first instance of such a mis-speculation or a certain number of mis-speculations must occur before data speculation can be disabled.

WARF’s proposed construction for “prediction” should be adopted because it captures how the term is defined in the claim and how it is used in the specification, especially in the context of the *second tier* of the invention. Intel’s proposal defines the term on the basis of one embodiment disclosed in the ’752 patent and by incorrectly equating prediction to “a mis-speculation counter.”

IV. "PREDICTION ASSOCIATED WITH THE PARTICULAR DATA CONSUMING INSTRUCTION AND BASED ON THE MIS-SPECULATION INDICATION"

Claim Term	WARF's Proposal	Intel's Proposal
prediction associated with the particular data consuming instruction and based on the mis-speculation indication	a dynamic multi-bit value associated with the particular LOAD instruction which indicates the likelihood that the data speculative execution of that LOAD instruction will result in a mis-speculation, and which is based on historical mis-speculation indications.	a value associated with the particular load instruction that indicates the likelihood that data speculative execution of a load/store pair will result in a mis-speculation, and which is based on mis-speculation indications.

The proper construction of this phrase follows from the construction of "prediction" as "a dynamic multi-bit value which indicates the likelihood that the data speculative execution of a LOAD instruction will result in a mis-speculation," for the reasons stated above in Section III, and the plain language of Claim 1. Again, there is no reference in this limitation to load/store pairs; indeed, stores are not even mentioned. While Claim 1 requires "a mis-speculation indication," as noted above, it is well established that in conventional patent parlance "a" indicates "one or more." *See, e.g., Baldwin Graphic,* 512 F.3d at 1342. Indeed, the specification of the '752 patent describes an embodiment wherein a first mis-speculation indication is provided which updates the prediction with further mis-speculation indications updating the prediction over time. Umberger Decl. Ex. 1, Col. 12, l. 64-Col. 13, l. 3; *see also* Dally Supp. Decl. ¶ 17.

V. "PREDICTOR"

Claim Term	WARF's Proposal	Intel's Proposal
predictor	a circuit that receives a mis-speculation indication from the data speculation circuit to produce a prediction.	a circuit that receives a mis-speculation indication from the data speculation circuit to produce a prediction based on historical mis-speculation indications.

In an attempt to import a limitation into Claim 1, Intel argues that the "predictor" should be explicitly limited to a circuit that is "based on historical mis-speculation indications." The claim language, the specification, and the file history, however, make clear that the "predictor" is a circuit that produces a prediction, and the prediction (and not the predictor) is based on one or more mis-speculation indications, i.e., historical mis-speculation indications.

The plain reading of Limitation (a) is in accord with WARF's proposed construction. That limitation requires a "predictor" that produces a prediction. The prediction in turn is (1) associated with a particular LOAD, and (2) based on one or more mis-speculation indications. Thus, the prediction, and not the predictor, is based on historical mis-speculation indications.

The specification of the '752 patent also supports WARF's proposed construction. The Summary of the '752 patent refers to "a predictor circuit . . . to produce a prediction . . . based on the mis-speculation indication." Umberger Decl. Ex. 1, Col. 4, ll. 17-21. The specification later mirrors a similar nexus between predictor, prediction and mis-speculation: "The **prediction provided by the predictor circuit** 33 . . . **is updated based on historical mis-speculations** detected by the data speculation circuit 30." Umberger Decl. Ex. 1, Col. 8, ll. 7-9 (emphasis added). Intel notes in its brief that in the *second tier* "a predictor based on the past history of mis-speculations for that LOAD instruction is employed to determine whether the instruction should be executed or delayed." Umberger Decl. Ex. 1, Col. 4, ll. 1-4. In light of the entire disclosure of the patent, however, the correct interpretation of the phrase is that the employed predictor produces a prediction leveraging the past history of the LOAD instruction.

Finally, Intel's reliance on the file history to limit the "predictor" circuit is to no avail. When distinguishing the claimed "predictor" from the predictor of the Hinton patent, the '752 inventors stated (emphasis added):

> The Applicant can find no indication in Hinton of a **predictor that receives a mis-speculation indication to produce a prediction as is claimed in claim 1**.

'752 File History (filed with the Court as Dkt. No. 28) at W0000262. In other words, the file history underscores the language of Claim 1 that the predictor produces a prediction, which in turn is based on a mis-speculation indication. Intel's reliance on selective quotes instead of complete sentences from the file history presents an incomplete picture of the predictors in the '752 and the Hinton patents. When one reads the entire paragraph from the patentees' response in the file history, it is clear that inventors argued that the predictor of the Hinton patent predicts by examining ready instructions whereas the "predictor" of the '752 patent predicts by examining previous instances of mis-speculation. *Id.* The file history does not indicate that the predictor, a circuit, is based on a mis-speculation indication, which is an event.[22]

VI. "MIS-SPECULATION"

The language of the preamble of Claim 1 defines mis-speculation as instances of data mis-speculation (emphasis added):

> 1. In a processor capable of executing program instructions in an execution order differing from their program order, the processor further having a data speculation

[22] Importing limitations from the file history is particularly disfavored; "this occurs only when the applicant makes "a *clear and unmistakable* disavowal of scope." *Purdue Pharma L.P. v. Endo Pharms., Inc.*, 438 F.3d 1123, 1136 (Fed. Cir. 2006) (emphasis added). *See also Phillips*, 415 F.3d at 1317 (the prosecution history "often lacks the clarity of the specification and thus is less useful for claim construction purposes").

circuit for detecting data dependence between instructions and detecting a **mis-speculation** where a [LOAD instruction] dependent for its data on a [STORE instruction] of earlier program order, is in fact executed before the [STORE instruction], a data speculation decision circuit comprising:

Ignoring this express definition as well as the specification, Intel proposes a construction that would include types of mis-speculation that were expressly distinguished from the claimed invention:

Claim Term	**WARF's Proposal**	**Intel's Proposal**
mis-speculation	where a LOAD instruction, that is dependent for its data on a STORE instruction appearing earlier in program order, is in fact executed before the STORE instruction.	where an instruction has been in fact executed prematurely and erroneously.

A. Two Flavors Of Speculation: Control And Data

The Background section of the ’752 patent states that a processor can perform both control speculation and data speculation. Umberger Decl. Ex. 1, Col. 2, ll. 26-33. Control speculation involves executing an instruction that follows a branch instruction by predicting the outcome of that branch. If the branch was not taken then the instruction was properly executed. Thus, control speculation involves a guess whether a branch would be taken or not. Umberger Decl. Ex. 1, Col. 1, ll. 64-66; Col. 2, ll. 33-36. Data speculation involves executing a LOAD instruction even though there are earlier STORE instructions that may change the value of the data that the LOAD seeks to access. Thus, data speculation involves a guess whether the earlier STORE instructions will alter the data that LOAD seeks, i.e., whether a data dependency exists or not. Umberger Decl. Ex. 1, Col. 1, l. 67-Col. 2, l. 5; Col. 2, ll. 36-40.

In either type of speculation, the processor makes a guess. During execution, a processor “may execute some dependent instructions before the instructions on which they are dependent.” Umberger Decl. Ex. 1, Col. 2, ll. 44-46. Eventually, the processor verifies whether the guess was correct or not. If either type of speculation was incorrect, "then the results of the prematurely executed dependent instructions must be discarded ('squashed')." Umberger Decl. Ex. 1, Col. 2, ll. 46-50. Thus, the moniker that an instruction has been "prematurely executed" applies to errors arising from either *control* speculation or *data* speculation.

B. Claim 1 Defines Mis-speculation As Data Mis-speculation

Even though a processor is capable of both control and data speculation, the '752 patent does not address control speculation because "[c]ontrol speculation is well understood in the art" and the "points w[h]ere control speculation is needed are clearly identified." Umberger Decl. Ex. 1, Col. 3, ll. 26-28. The '752 inventors tackled data speculation because:

> In contrast [to control speculation] the points where data speculation is needed are not clear since any instruction loading data from memory can be data dependent on any previous instructions that writes to memory. Consequently, predicting and tracking data dependencies, "data dependence speculation" can easily become overwhelming.

Umberger Decl. Ex. 1, Col. 3, ll. 39-43. It is evident that the '752 invention attempts to solve the problems hindering the use of data speculation and not control speculation. As a result, the preamble to Claim 1 requires a "data speculation circuit" capable of detecting a mis-speculation. And, a mis-speculation occurs

> where a data consuming [LOAD] instruction dependent for its data on a data producing [STORE] instruction of earlier program order, is in fact executed before the data producing [STORE] instruction[.]

Umberger Decl. Ex. 1, Claim 1.

Thus the express language of Claim 1 narrows mis-speculation to instances of data mis-speculation, consistent with numerous references in the specification.[23] *See, e.g.*, Umberger

---

[23] The parties agree that data speculation is "execution of a LOAD instruction that may have a data dependency on a STORE instruction appearing earlier in program order as if the LOAD instruction had no data dependency at all." Umberger Decl. Ex. 2, at 3. According to

(Footnote continued)

Decl. Ex. 1, Col. 3, ll. 38-48 and ll. 63-65 (referring to "data mis-speculation"); Col. 4, ll. 11-14, 33-37 and 47-50 (same); Col. 6, ll. 51-59; Col. 9, l. 39-Col. 10, l. 55 (describing operation of data speculation circuit); *see Phillips*, 415 F.3d at 1316 (patentee may give a "special definition" to a claim term and "[i]n such cases, the inventor's lexicography governs"). When "the claim itself contains a precise definition," a district court may rely "heavily on the claim language" for construction. *Tip Systems, LLC v. Phillips & Brooks/Gladwin, Inc.*, Nos. 2007-1241, 1279, U.S. App. LEXIS 12757, at *8, 529 F.3d 1364 (Fed. Cir. June 18, 2008).

WARF's construction adopts the definition provided in the claim language and simplifies it to make it more accessible to the jury. Dally Supp. Decl. ¶¶ 23-25; *Phillips*, 415 F.3d at 1312 ("It is a bedrock principle of patent law that the claims of a patent define the invention to which the patentee is entitled the right to exclude.") (internal quotation marks omitted); *John Mezzalingua Assocs.*, 298 F. Supp. 2d at 817 ("resort must be had in the first instance to the words of the claim") (internal quotation marks omitted).

C. Intel's Selective Use Of The Specification And The Preamble Is Improper

Intel's proposed construction is incorrect on several grounds: *first*, it does not correctly reflect the intended scope of the term; *second*, it is based on an incomplete reading of the specification; and *third*, it inexplicably omits highly relevant portions of Claim 1.

By defining "mis-speculation" by its consequence, Intel's definition captures both data and control speculation. As mentioned above, an instruction is deemed to have been "prematurely executed" regardless of whether the erroneous speculation was control or data. Thus, Intel's reliance on the result to define the event leads to an imprecise construction. In

this definition, data mis-speculation is execution of a load instruction before a store instruction where the load is dependent upon the store instruction occurring earlier in program order.

contrast, WARF's definition focuses on the event that constitutes data mis-speculation. The claim language unambiguously defines mis-speculation to refer to data mis-speculation. Dally Supp. Decl. ¶ 26.

Though Intel claims that its construction "adheres precisely to the specification," Intel Markman Br. at 35, a closer inspection reveals that Intel omits a significant portion of the specification. The complete sentence, which Intel quotes only in part, and the sentence prior to that reads as follows:

> Prior to the retirement circuit 26 writing values to memory, a data speculation circuit 30 communicating with the allocation circuit 20 and the retirement circuit 26 detects mis-speculation. **<u>As described above</u>**, mis-speculation occurs in an instruction that has been executed prematurely and erroneously.

(Umberger Decl. Ex. 1, Col. 7, ll. 36-41 (emphasis added)). Intel conveniently ignores the "[a]s described above" portion of the sentence. Because the specification expressly refers to the preceding disclosure, review of such information is warranted. In the preceding sentence, the '752 patent mentions that the "mis-speculation" is detected by a **data** speculation circuit, thereby implying that the mis-speculation is of the **data** kind. Similarly, earlier in the same section, the '752 patent states that "it is important to identify and address ambiguous **data dependencies to eliminate mis-speculation** and time consuming instruction squashing." Umberger Decl. Ex. 1, Col. 6, ll. 55-59 (emphasis added); *see also* Col. 7, ll. 4-7 (describing that the data speculation circuit track memory operations (i.e., data operations) to detect mis-speculations).

Finally, Intel picks the "in fact executed" phrase from the definition of "mis-speculation" from the claim but omits the remainder of the definition. In its brief, Intel does not provide any reason why the definition from the claim should be ignored. Moreover, addition of this term raises more problems. Intel has offered a controversial definition of what it means to "in fact execute" a LOAD. Intel Markman Br. at 23-27. However, it does not explain what it means to "in fact execute" other types of instructions. Intel's brief describes at least three types of instructions: computation instructions, data instructions, and control instructions. Intel Markman Br. at 2-3. All of these can be executed in a speculative manner and can result in a mis-speculation.

Although Intel advocates deleting "program order" and references to LOAD and STORE instructions from the claim language, it does not offer any justification for doing so. The claim language expressly refers to LOAD and STORE instructions because it pertains to data mis-speculations, which occur when a LOAD that appears after a STORE in "program order" and depends on that STORE is executed ahead of that STORE. Without a corresponding concept in the proposed definition or a valid justification, it would be an error to adopt a construction that omits claim terms.

In conclusion, WARF's definition is the correct one because it relies on the claim language to explain what mis-speculation correctly implies within the bounds of the '752 patent.

VII. "MIS-SPECULATION INDICATION"

Claim Term	**WARF's Proposal**	**Intel's Proposal**
mis-speculation indication	an indication that the data speculative execution for a LOAD instruction was incorrect.	an indication that an instruction has been executed prematurely and erroneously

The parties' dispute regarding this term originates from their dispute regarding the meaning of "mis-speculation." Based on WARF's argument that the '752 patent narrows "mis-speculation" to data speculation, a "mis-speculation indication" is an indication that the data speculation execution of a LOAD instruction was incorrect. A lack of such indication would imply that such an execution for that LOAD instruction was correct. The "prediction" associated with a LOAD instruction is based on the presence and absence of one or more of such mis-speculation indications.

VIII. "IN FACT EXECUTED"

Claim Term	WARF's Proposal	Intel's Proposal
where a data consuming instruction dependent for its data on a data producing instruction of earlier program order, is **<u>in fact executed</u>** before the data producing instruction	a LOAD instruction is "in fact executed" before the STORE instruction when the LOAD instruction has actually accessed or was certain to access data that has not yet been updated by the STORE instruction.	a load instruction is "in fact executed" when the load instruction actually has loaded data from a memory location

A. The Data Speculation Circuit Tracks "In Progress" LOAD Instructions

The dispute regarding this phrase primarily boils down to how the '752 patent handles in-progress LOAD instructions. While Intel argues that the circuit is limited to detecting mis-speculations only for those LOAD instructions that have "actually loaded data from a memory location," the data speculation circuit in the '752 patent tracks in-progress LOAD instructions in order to detect data mis-speculations.

Because the phrase "in fact executed" does not have a generally accepted and well understood meaning, it is essential to understand the context in which the phrase occurs. Dally Supp. Decl. ¶ 37. The preamble of Claim 1 makes clear that the term pertains to (i) detecting a data mis-speculation, (ii) which is detected by a data speculation circuit (iii) located in a processor capable of executing instructions out of order.

As WARF pointed out on pages 36-37 of its opening brief, the '752 patent monitors in-progress LOAD instructions in order to detect data mis-speculations. The data speculation circuit is "responsible of keeping track of order of the memory operations as they are being performed by the processing units so that it can detect any mis-speculations." Umberger Decl. Ex. 1, Col. 7, ll. 3-7. The processor informs the data speculation circuit of the identity of memory operations (i.e., LOAD and STORE instructions) before the LOAD instruction "actually has loaded data from a memory location." In fact, the '752 patent provides:

> Prior to reading data from memory or requesting a store operation the processing units 24 notify the data speculation unit 30 of the operation so that the latter can, in conjunction with the allocation unit, keep track of the program and execution order of the operations.

Umberger Decl. Ex. 1, Col. 7, ll. 25-30. Thus, the data speculation circuit begins tracking the LOAD instruction before it actually has loaded data. Blocks 48-52 of Figure 3 depict that for every STORE instruction, the circuit "checks other **concurrent** LOAD instructions to see if they have been prematurely executed and thus whether there has been a mis-speculation." Umberger Decl. Ex. 1, Col. 9, ll. 61-64 (emphasis added). Because the data speculation circuit starts tracking a LOAD instruction before it "actually has loaded data," concurrent LOAD instructions encompass LOAD instructions that (i) have actually accessed data or (ii) are in the process of accessing data that has not yet been updated by the STORE instruction. If a LOAD instruction has either accessed incorrect data or is certain to do so, then such a LOAD is deemed to have mis-speculated. Dally Supp. Decl. ¶¶ 30-32.

Intel's requirement that the LOAD should have loaded data prior to detecting a mis-speculation is artificial because mis-speculation can be detected before the data is actually loaded. Perhaps driven by this mistake, Intel repeatedly articulates what is means to execute a LOAD instruction in the context of detecting a mis-speculation:

- "A load instruction is actually carried out ('in fact executed') when the instruction loads the value from the designated memory location." Intel Markman Br. at 24.

- "Thus, a load instruction executes when it 'loads the contents' of a specified 'memory location.'" Intel Markman Br. at 25.

As the context makes clear, however, the issue is not whether a LOAD has in fact executed. Rather this issue is whether a LOAD has in fact executed before a STORE it depends upon. Simply put, whether there has been a mis-speculation between a LOAD and a STORE.

The '752 patent clearly teaches that a data mis-speculation can be detected before a LOAD instruction "actually has loaded data from a memory location." In fact, whether a LOAD instruction conflicts with a STORE instruction can be verified as soon as the address the LOAD *will* read from and the address the STORE *will* write to are known. If the two addresses are the same and the LOAD occurs after the STORE, then executing the LOAD ahead of the STORE will result in a data mis-speculation. Umberger Decl. Ex. 1, Col. 6, ll. 18-54 (explaining that dependence between LOAD and STORE instructions can be detected once the addresses these instructions are operating on are known); *see also* Intel Markman Br. at 8-10 (referring to the same portion of the '752 patent). Thus, in the "data speculation circuit," detection of mis-speculation can begin once the addresses referred to by the LOAD and the STORE are known. If a comparison with a STORE indicates that the LOAD instruction is certain to access data that has not yet been updated by the STORE instruction (i.e., an error is going to occur), then common sense dictates that corrective action should be initiated as soon as possible. Dally Supp. Decl. ¶¶ 33 and 34.

B. The Appropriate Context To Understand "Executed" Is An Out-Of-Order Processor Capable Of Data Speculation

Intel leans on various dictionaries to argue that in Claim 1, "executed" means that an instruction "is finished with the process of execution," i.e., execution implies completion. Intel Markman Br. at 24-25. Intel is again incorrect on several grounds.

First, Intel's scouring of dictionaries to ascertain what it means to execute a LOAD instruction is futile because "execute" does not have a universally applicable meaning. Dally Supp. Decl. ¶ 37. Intel's collection underscores WARF's argument: to execute means to carry out, perform or run at least (i) an instruction, (ii) a command, (iii) a function, (iv) a process, or (v) even a computer program. Intel Markman Br. at 24. None of these definitions even hint at what it means to execute a LOAD before a STORE in the context of detecting a data mis-speculation in a out-of-order processor. *Second*, the cited dictionaries simply define "execute" as a process of carrying out an instruction. *Id.* They do not state that "executed" means an instruction has completed execution. In an out-of-order processor, a instruction may be "executed" and squashed prior to completion. Dally Supp. Decl. ¶ 38. In that case, the instruction has still "executed" though only partially. *Third*, the proposed definition contradicts how a person of ordinary skill would interpret the term. A skilled person would understand "executed" to mean that execution occurred in the past. It does not necessarily imply completion. Dally Supp. Decl. ¶ 39.

While there are instances where reliance on dictionaries is appropriate, "in fact executed" is not one of those. The proper construction for a disputed claim term is not the "abstract meaning of [a] word[]" found in a dictionary; it is the meaning that a claim term would have to a person of ordinary skill in the art read "*in the context of the entire patent*." *Phillips*, 415 F.3d at 1311; *id.* at 1321 (it is error to "adopt[] . . . a dictionary definition entirely divorced" from the specification and the context of the invention) (emphasis added).

To the extent the Court must rely on extrinsic evidence to ascertain the meaning of "in fact executed," it must do what a person of ordinary skill would do – review technical literature contemporaneous to the '752 patent that discuss data mis-speculation in out-of-order processors.

Dally Supp. Decl. ¶ 37. One such reference is U.S. Patent No. 5,751,983 entitled "Out-of-Order Processor With A Memory Subsystem Which Handles Speculatively Dispatched Load Operations", which was filed in October 1995 (more than a year before the '752 patent was filed) and assigned to Intel. Umberger Decl. Ex. 5; Dally Supp. Decl. ¶ 40. The '983 patent pertains "particularly, to speculative execution of operations in computer systems." Umberger Decl. Ex. 5, Col. 1, ll. 7-9. According to the Abstract, the patent teaches a "method and apparatus for speculatively dispatching and/or executing LOADs in . . . a [sic] out-of-order processor that handles LOAD and STORE operations . . ." in order to prevent "[m]is-speculated loads" from "corrupting . . . the machine with invalid data." Umberger Decl. Ex. 5, Abstract.

The Memory Order Buffer (MOB) employed in the Intel patent bears a striking resemblance to the "data speculation circuit" in regard to the detection of data mis-speculation. The data speculation circuit tracks memory operations in order to detect mis-speculations. Similarly, the MOB "controls dispatching, buffers STORE . . . and LOAD operations, tracks memory operation progress through execution and the retirement process." Umberger Decl. Ex. 5, Col. 7, ll. 58-61. "Once a LOAD operation has been dispatched", the MOB, in a manner similar to the data speculation circuit, "looks to detect one or more conflicts." Umberger Decl. Ex. 5, Col. 9, l. 65 – Col. 10, l. 1. Thus, instead of dictionaries that capture generic meanings, the '983 patent is more likely to address execution of LOAD instructions in the appropriate context. Dally Supp. Decl. ¶ 40.

In fact, Intel's '983 patent discusses detecting data speculation in out-of-order processors. According to the patent, "when a LOAD operation is ready to execute . . . the LOAD operation is sent to data cache 205 [to retrieve data] and assigned a valid and completed LOAD status. It should be understood that this does not necessarily mean that the LOAD operation has been executed." Umberger Decl. Ex. 5, Col. 11, ll. 45-51. In other words, in Intel's processor, even though the LOAD operation has **<u>not</u>** been executed, it is considered completed or carried out.

To unravel the conundrum, Intel explained in 1995 (which ironically WARF iterates today) that the meaning of to have executed a LOAD instruction depends purely upon the perspective:

> With respect to the writeback operation to RS 305 and ROB 306 [i.e., other elements of the processor], the LOAD operation is not considered completed because there may have been a cache miss. But the LOAD is considered completed from the perspective of MOB 503 and no further action needs to be taken. . . . **The LOAD may <u>not</u> have actually happened, but as far as MOB 503 is concerned the operation is complete**.

Umberger Decl. Ex. 5, Col. 11, ll. 51-56, 59-61 (emphasis added). Thus, from the perspective of the MOB in the '983 patent, Intel considers the LOAD to have "in fact executed" (i.e., completed or carried out) even though the LOAD has not yet loaded the data. Dally Supp. Decl. ¶ 40. The same should be true from the perspective of the data speculation circuit in the '752 patent. Dally Supp. Decl. ¶ 41. By arguing now that "executed" implies completion by a LOAD instruction in the context of data mis-speculation, Intel clearly contradicts itself.

Mirroring Intel's explanation from 1995, the meaning of "in fact executed" should be interpreted from the perspective of the data speculation circuit (or the MOB), which is <u>whether the execution of a LOAD has progressed to a point that the circuit can declare a data mis-speculation</u>. It is not necessary for the LOAD to have loaded the data in order to detect a data mis-speculation. This implicates LOAD instructions in **two** stages of execution: those that have accessed and those that are certain to access the not yet updated data. Dally Supp. Decl. ¶¶ 30-32 and 34. Only WARF's construction captures both of these situations.

Finally, WARF's construction reflects how out-of-order processors in the industry were detecting data mis-speculation prior to the filing of the '752 patent. Several patents indicate that out-of-order processors in that time frame uniformly detected data mis-speculation by tracking LOAD instructions that had been completed or were still in progress. Dally Supp. Decl. ¶¶ 40-42 (discussing U.S. Patent No. 5,751,983 to Abramson et al. (Umberger Decl. Ex. 5); U.S. Patent No. 5,931,957 to Konigsburg et al. (Umberger Resp. Decl. Ex. 1)).

C. "Memory Location" Refers To Multiple Locations In The Memory Hierarchy

Intel argues that because a LOAD instruction operates on a memory location, the definition should expressly state that the LOAD instruction loads data "from a memory location." Intel Markman Br. at 27. It would be error to adopt Intel's definition because that definition does not clarify what that memory location is. Dally Supp. Decl. ¶ 35.

Modern high-performance processors employ memory hierarchy to improve performance of LOAD and STORE instructions. Dally Decl. ¶¶ 24-20. The hierarchy may consist of a Store Buffer, multiple levels of Cache memories, and a Main memory.[24] *Id.* A LOAD instruction may load data from a Main memory location, a Cache location or a Store Buffer location. Dally Supp. Decl. ¶ 35. By failing to clarify whether the usage of "memory location" refers to one or all of these possibilities, Intel's definition has a potential of creating considerable confusion. For example, if "memory location" refers to only a location in the Main memory, then the LOAD instructions that load their data from the Cache or from the Store Buffer would be improperly excluded from Intel's definition. Dally Supp. Decl. ¶¶ 35-36.

[24] The concept of memory hierarchy using caches and main memory was well known before the '752 patent was filed. In fact, Intel's expert, Dr. Clark published a paper, titled "Design issues and tradeoffs for write buffers," wherein he refers to processors with cache memories from pre-1996 timeframe. *See, e.g.*, Umberger Resp. Decl. Ex. 3, Abstract and cited reference nos. 9, 10, and 13.

In contrast, WARF's definition avoids this morass by focusing on what matters when detecting data mis-speculation – whether a LOAD has accessed or is certain to access data before is has been updated by the STORE. WARF's proposed construction interprets the term not in isolation but in context of the claim, which is detecting a data mis-speculation. Though the precise words used to explain the arcane technical concepts hidden in the term "in fact executed" may not expressly appear in the specification, they are nevertheless disclosed in the specification and reflect how the industry (including Intel) understands and uses that term.

IX. "WHERE A DATA CONSUMING INSTRUCTION DEPENDENT FOR ITS DATA ON A DATA PRODUCING INSTRUCTION OF EARLIER PROGRAM ORDER, IS IN FACT EXECUTED BEFORE THE DATA PRODUCING INSTRUCTION"

Claim Term	**WARF's Proposal**	**Intel's Proposal**
where a data consuming instruction dependent for its data on a data producing instruction of earlier program order, is in fact executed before the data producing instruction	where a LOAD instruction, that is dependent for its data on a STORE instruction appearing earlier in program order, has actually accessed or was certain to access data that has not yet been updated by the STORE instruction	where a load instruction that depends on a store instruction has actually loaded data from a memory location before the store instruction has put data into that same memory location

As the preamble to Claim 1 clearly indicates, this phrase articulates what constitutes a **data mis-speculation**. The dispute regarding this phrase continues from WARF's and Intel's competing views regarding "in fact executed" in the context of a data mis-speculation. In addition to Intel's errors outlined in preceding section, Intel's proposal for the larger phrase suffers from two additional infirmities.

First, as the claim language indicates, a data mis-speculation occurs (i) when a LOAD that uses data updated by a STORE executes ahead of that STORE and (ii) the STORE occurs earlier than the LOAD in program order. Thus, the claim requires that both of these conditions must be satisfied before a data mis-speculation can be declared. Intel does not offer any reason why a constraint expressly imposed by the claim language should be omitted, especially given the Federal Circuit's instruction that "claims are interpreted with an eye toward giving effect to all terms in the claim." *See Bicon, Inc. v. Straumann Co.*, 441 F.3d 945, 950 (Fed. Cir. 2006) ("Allowing a [party] to argue that physical structures and characteristics specifically described in a claim are merely superfluous would render the scope of the patent ambiguous, leaving examiners and the public to guess about which claim language the drafter deems necessary to his claimed invention and which language is merely superfluous, nonlimiting elaboration."). WARF's proposal maintains that requirement and, to assist a lay person, expands how a person of ordinary skill would interpret the term "in fact executed" in this context.

Second, Intel's proposal requires that for a mis-speculation to occur, the LOAD should have "actually loaded data from a memory location" before the STORE places it in the "same memory location." This construction is not attuned to how processors and memory hierarchies worked even prior to the filing of the '752 patent. Dally Supp. Decl. ¶¶ 30 and 31. As explained above, by failing to clarify what "memory location" implies, Intel's definition creates confusion. Moreover, a mis-speculating LOAD instruction can be in progress or could have already fetched the data when a data mis-speculation is detected. Dally Supp. Decl. ¶¶ 35 and 36. To mis-speculate, a LOAD instruction is not required to have "actually loaded data from a memory location." It is enough for a LOAD to be on an irreversible course to obtain incorrect data and thereby cause an error. Dally Supp. Decl. ¶ 34.

WARF’s construction should be adopted because it remains faithful to the claim language and reflects the understanding of a person of ordinary skill, which Intel’s definition fails to do.

CONCLUSION

In its attempt to import restrictions and limitations into independent Claim 1, Intel has presented this Court with a fundamental mischaracterization of the invention disclosed in the ’752 patent. WARF has demonstrated the error of Intel’s view of the patent and its proposed claim constructions. In response to the chart presented by Intel in its opening brief at 18, WARF provides the following accurate description of the invention recited in Claim 1:

’752 Patent, Claim 1	**“Plain English” Summary**
In a processor capable of executing program instructions in an execution order differing from their program order, the processor further having a **<u>data speculation circuit</u>** for detecting data dependence between instructions and detecting a mis-speculation where a [LOAD instruction] dependent for its data on a [STORE instruction] of earlier program order, is in fact executed before the [STORE instruction], a data speculation decision circuit comprising:	A processor that is capable of out of order execution of instructions and contains a <u>data speculation circuit</u> that detects data dependence between LOAD and STORE instructions and tracks their execution in order to detect data mis-speculations a <u>data mis-speculation</u> occurs when a LOAD instruction is in fact executed before a STORE instruction and where (1) the LOAD instruction uses the data that the STORE instruction updates and (2) the STORE instruction is supposed to update the data before the LOAD instruction uses it a LOAD is <u>in fact executed</u> before a STORE when the LOAD has actually accessed or is certain to access stale data, i.e., data that has not yet been updated by the STORE instruction
a) a predictor receiving a mis-speculation indication from the data speculation circuit to produce a prediction associated with the [LOAD instruction] and based on the mis-speculation indication; and	A <u>predictor</u> is informed whenever data speculative execution of a LOAD instruction turns out to be an error, which in turn produces a <u>prediction</u> that is associated with a LOAD instruction The <u>prediction</u> is based on the outcomes (i.e., correct or incorrect) of past data speculative executions of the LOAD instruction, as a result, <u>prediction</u> provides the likelihood whether the next data speculative execution of the LOAD instruction will result in an error Before issuing a LOAD instruction, the prediction associated with that LOAD is obtained
b) a prediction threshold detector preventing data speculation for instructions having a prediction within a predetermined range.	When the prediction for the LOAD instruction is “within a predetermined range” (i.e., indicates that the data speculative execution will result in a mis-speculation), the LOAD is delayed (i.e., data speculation is prevented for that instruction)

Dated this 24th day of July, 2008.

HELLER EHRMAN LLP

By: *__/s/ Michelle M. Umberger__*
John S. Skilton, SBN 1012794
John.Skilton@hellerehrman.com
Michelle M. Umberger, SBN 1023801
Michelle.Umberger@hellerehrman.com
Gabrielle E. Bina, SBN 1041749
Gabrielle.Bina@hellerehrman.com
Lissa R. Koop, SBN 1050597
Lissa.Koop@hellerehrman.com
One East Main Street, Suite 201
Madison, WI 53703-5118
Telephone: (608) 663-7460
Facsimile: (608) 663-7499

Robert T. Haslam
Robert.Haslam@hellerehrman.com
Anupam Sharma
Anupam.Sharma@hellerehrman.com
275 Middlefield Road
Menlo Park, CA 94025
Telephone: (650) 324-7000
Facsimile: (650) 324-0638

Attorneys for Wisconsin Alumni Research Foundation